IPG Politécnico da Guarda
Polytechnic of Guarda

Mestrado em Computação Móvel

Sistema de monotorização e controlo de sistema solar - SIMONIC

Luís Miguel G. Almeida Tavares Saraiva

julho | 2016

Escola Superior
de Tecnologia e Gestão

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

SIMONIC

SISTEMA DE MONITORIZAÇÃO E CONTROLO DE SISTEMA SOLAR

Estágio Profissionalizante do Mestrado em

Computação Móvel

Luís Miguel Gonçalves Almeida Tavares Saraiva j

julho | 2016

Escola Superior de tecnologias e Gestão

Instituto Politécnico da Guarda

SIMONIC

SISTEMA DE MONITORIZAÇÃO E CONTROLO DE SISTEMA SOLAR

Relatório do Estágio Profissionalizante submetido como requisito parcial

Para obtenção do grau de Mestre em Computação Móvel

Orientador: Professor Doutor Adérito Neto Alcaso

Coorientador: Professor Doutor Paulo Vieira

Supervisor: Professor Carlos Alberto Figueiredo Ramos

Estágio Profissionalizante do Mestrado em Computação Móvel

Luís Miguel Gonçalves Almeida Tavares Saraiva

julho | 2016

Agradecimentos

Os meus mais sinceros agradecimentos a todos aqueles que tornaram a realização deste trabalho possível.

Gostaria antes de mais de agradecer ao meu orientador Professor Doutor Adérito Neto Alcaso, coorientador Professor Doutor Paulo Vieira e Supervisor Professor Carlos Alberto Figueiredo Ramos pelo apoio, incentivo e disponibilidade demonstrada em todas as fases que levaram à concretização deste trabalho.

Agradeço ao Instituto Politécnico da Guarda (IPG) e ao Centro de Investigação em Sistemas Electromecatrónicos (CISE) que tornaram possível este estágio.

Por último, quero agradecer à minha família pelo apoio incondicional ao longo destes anos.

Resumo

Este relatório insere-se no âmbito de um estágio profissionalizante do mestrado em computação móvel realizado na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico da Guarda (IPG). Neste estágio desenvolveu-se um projeto, designado por SiMoniC, de **S**istema de **M**onitorização e **C**ontrolo, para um sistema de conversão de energia baseado no sol e conhecido por painel termofotovoltaico (PVT). O projeto foi implementado com recurso a duas plataformas microcontroladoras embebidas, sistemas de transmissão sem fios (*wireless*) e à computação na internet (vulgarmente conhecida por nuvem ou *cloud*). O principal objetivo do projeto é o de disponibilizar o acesso remoto e em tempo real dos dados monitorizados (corrente elétrica, tensão elétrica, temperatura à entrada do fluido, temperatura de saída do fluido, temperatura superior do painel PVT, temperatura inferior do painel PVT, temperatura da parte de trás do painel PVT, temperatura ambiente, radiação solar, velocidade do vento, direcção do vento e caudal do fluido hidráulico), assim como possibilitar algum controlo ao funcionamento do sistema. Para tal, foram desenvolvidos dois métodos, o primeiro utiliza o serviço *cloud* da Google e o segundo utiliza um servidor web de uma das plataformas. Com este projeto demonstra-se a viabilidade do uso de plataformas microcontroladoras de baixo custo e de serviços internet grátis existentes na Web, para apoio ao estudo remoto de sistemas de energias renováveis, dispensando a aquisição de sistemas dedicados, normalmente mais caros e limitados no tipo de processamento que propõem.

Palavras-chave

PVT, Sistemas embebidos, Transmissão sem fios, Computação na nuvem, Servidor web.

Abstract

This report is part of the framework of a stage experience of master's degree in mobile computing, held at the Superior School of Technology and Management (ESTG) of the Polytechnic Institute of Guarda (IPG). In this stage a project called SiMoniC, for **S**ystem **M**onitoring and **C**ontrol, was developed for an energy conversion system based on the sun and known as thermophotovoltaic panel (PVT). The project was implemented using two embedded microcontrollers platforms, wireless transmission systems and net computing (commonly known as cloud). The main objective of the project is to provide remote access and real-time data monitoring (like: electrical current, electrical voltage, input fluid temperature, output fluid temperature, backward PVT temperature, top PVT glass temperature, bottom PVT glass temperature, ambient temperature, solar radiation, wind speed, wind direction and mass fluid flow), so as to allow some control to the operation of the system. To this end, two methods were developed, the first uses Google cloud service and the second uses a web server based in one of the platforms. This project demonstrates the feasibility of using inexpensive microcontroller's platforms and free internet service in the Web, to support the remote study of renewable energy systems, eliminating the acquisition of dedicated systems typically more expensive and limited in the kind of processing proposed.

Keywords

PVT, Embedded systems, Wireless transmission, Cloud computing, Web server.

Índice

Índice de tabelas

Índice de figuras

Lista de abreviaturas

ADC: *Analog to digital converter* (Conversor Analógico digital)

BLUETOOTH: Padrão de comunicação sem fio de baixo consumo

CISE: Centro de Investigação em Sistemas Electromecatrónicos

CO_2 Dióxido de carbono

CURL: Ferramenta de linha de comando *open source* e uma biblioteca para transferir dados com a sintaxe URL

GIRS-RES: Estação Internacional de Investigação em Energias Renováveis

IDE: *Integrated Development Environment* (Ambiente de desenvolvimento integrado)

IP: *Internet Protocol* (Protocolo de internet)

IPG: Instituto Politécnico da Guarda

MD5: *Message-Digest algorithm* 5 (Algoritmo de encriptação "*hash*" de 128 bits unidirecional)

PHP: *Hypertext Preprocessor* (Linguagem de *script open source*)

PIB: Produto interno bruto

PVT: Painel termofotovoltaico

RAM: *Random Access Memory* (memoria de acesso aleatória)

SD CARD: Cartão de memória *Secure Digital*

SIMONIC: Sistema de monitorização e controlo de sistema solar

SMS: *Short Message Service* (serviço de mensagens curtas)

SQL: *Structured Query Language* (linguagem de consulta estruturada)

UHTTPD: Servidor web escrito de raiz pelos desenvolvedores OpenWrt / LuCI

URL: Uniform Resource Locator (endereço de um recurso disponível em uma rede)

XBEE: Tecnologia *wireless* de baixo custo da DIGI, desenvolvida como *open global standard*

Wi-Fi: Conjunto de especificações para redes locais sem fios (*WLAN - Wireless Local Area Network*) baseado no padrão IEEE 802.11.

ZIGBEE: Padrão de comunicações sem fio de baixo consumo, baixa transmissão de dados e baixo custos de implementação

Capítulo 1 – Introdução

Este trabalho integra a área das energias renováveis e das tecnologias de comunicação e computação e surgiu no âmbito do mestrado em computação móvel existente no Instituto Politécnico da Guarda, tendo como grande apelativo a junção de duas áreas em plena expansão, as energias renováveis e a computação móvel. O projeto foi desenvolvido nas instalações do Instituto Politécnico da Guarda (IPG), onde se encontra instalada a Estação Internacional de Investigação em Energias Renováveis (GIRS-RES) do Centro de Investigação em Sistemas Electromecatrónicos (CISE, 2016), pertencente ao sistema científico nacional enquadrado pela Fundação para a Ciência e Tecnologia.

1.1 Enquadramento

A monitorização de sistemas físicos é uma necessidade em diversos tipos de aplicações e situações, acompanhando a crescente automatização e controlo de processos e permitindo em muitas situações ultrapassar as limitações de medidas humanas. Nos últimos tempos, as áreas ambientais e energética têm aumentado a necessidade de monitorização de sistemas associados de forma a garantir a obtenção de objetivos definidos internacionalmente no que respeita às alterações climáticas e ao uso das energias renováveis. Entre estas destaca-se a energia solar, nas vertentes térmica e fotovoltaica, que pode dar um grande contributo para a satisfação das necessidades energéticas da humanidade. Contudo, enquanto a energia solar térmica tem já um potencial de exploração considerável, a energia solar fotovoltaica ainda precisa de evolução devido ao menor rendimento da conversão direta da radiação solar em eletricidade. Para maximizar esta conversão recorre-se a várias soluções, sendo uma delas a tecnologia híbrida solar térmica e fotovoltaica, implementada em painéis termofotovoltaicos, conhecidos por PVT.

A temática das energias renováveis e do clima está na ordem do dia, como a recente conferência de Paris demonstra, pelo que é pertinente um breve resumo do tema

1.1.1 Energias Renováveis

A relação do PIB e do consumo de energia estão diretamente ligados, ou seja quanto mais desenvolvido o país for, maior tende a ser o seu consumo energético. Um PIB elevado leva a um crescimento económico maior, que por sua vez aumenta o PIB e consequentemente o consumo de energia, como mostra a Tabela 1.

Tabela 1 - Os 10 Países com maior consumo de energia em 2014 (Enerdata 1, 2015)

Year: 2014 Unit: Mtoe	Highest ten
China	3,034
United States	2,224
India	872
Russia	751
Japan	437
Germany	307
Brazil	306
South Korea	277
Canada	251
France	243

Para sustentar esta cada vez maior necessidade de energia mundial, a grande fonte primária de energia tem sido baseada nos combustíveis fosseis, sendo que muita desta energia é usada na produção de eletricidade, como se mostra na Figura 1, para o caso dos Estados Unidos.

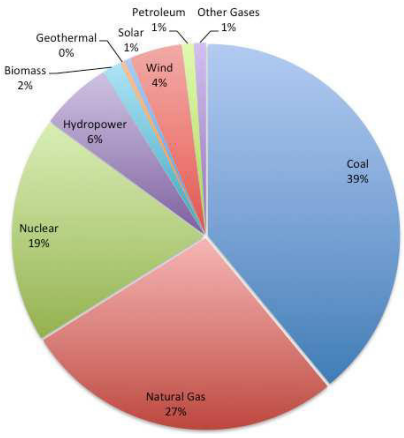


Figura 1 - Electricidade gerada por fonte em 2014 (Pace, 2015)

O aumento do aquecimento global é causado em grande parte pelas emissões de CO_2 derivados à queima de combustíveis fósseis. Visto a maior fonte de energia mundial derivar de combustíveis fósseis facilmente podemos concluir que as maiores potências mundiais são as que produzem maiores emissões de CO_2 (Tabela 2).

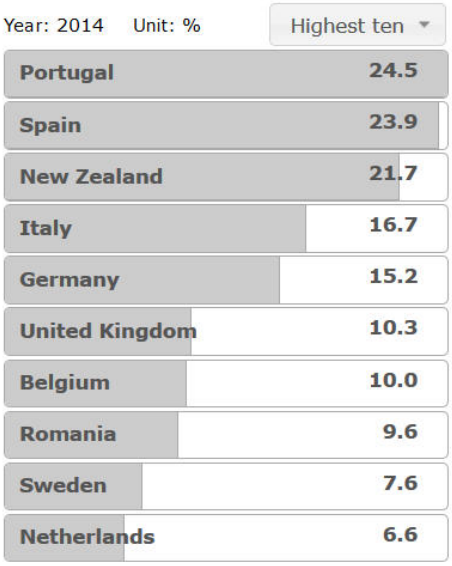
Tabela 2 - As emissões de CO_2 da queima de combustíveis fósseis por País em 2014 (Enerdata 2, 2015)

A alternativa aos combustíveis fósseis passa pelas energias "limpas" onde estão inseridas as energias renováveis. As energias renováveis caracterizam-se pela sua capacidade de se regenerar, sendo assim consideradas ilimitadas. Constituem energias renováveis a solar, eólica, hidráulica, biomassa, entre outras.

Portugal é um dos países a nível mundial que tem feito uma maior aposta nas energias renováveis. Sendo um país sem reservas de petróleo, grande parte da energia consumida é importada (aiceo Portugal Global, 2015). Com a aposta nas energias renováveis, Portugal conseguiu em 10 anos que cerca de 25 porcento da energia consumida seja produzida em Portugal com origem em energias renováveis (Pires, 2012). Em 2014 Portugal era um dos 10 países, a nível mundial, com maior quota de energias renováveis na produção de energia elétrica (Tabela 3) e com a maior contribuição solar nessa produção (Tabela 4).

Tabela 3 - 10 Países com maior quota de energias renováveis na produção de eletricidade (incluindo hidráulica) (Enerdata 3, 2015).

Tabela 4 - Os 10 países com maior quota de produção de energia solar e eólica na produção de energia (Enerdata 4, 2015).



O aproveitamento das energias renováveis, sendo intermitente e não controlável na fonte, necessita de sistemas de monitorização, nomeadamente climáticos, de forma a permitir prever e otimizar a conversão de energia, tornando este aproveitamento tão rentável quanto os tradicionais.

1.2 Motivação

A Figura 2 mostra uma imagem do sistema PVT instalado e do princípio da monitorização cablada. São visíveis os sensores ligados à medida das condições climatéricas assim como uma caixa de ligações onde estão colocados os sensores de corrente e tensão e os circuitos de condicionamento de sinal.

Figura 2 – Estrutura de sistema PVT

Os dados são transmitidos por cabo para um computador Pentium 4 com uma placa de aquisição DAQCard da National Instruments (National Instruments Corporation), sendo o seu tratamento, análise e armazenamento feitos com recurso a um programa desenvolvido em ambiente MatLab/Simulink. Entre o local das medidas e o computador existe uma distância de cerca de 15 metros. A visualização e consulta dos dados sobre o funcionamento do sistema apenas pode ser feita localmente, o que é um grande inconveniente. É também sabido que para estas distâncias são introduzidos erros na leitura dos sinais. A grande vantagem do sistema existente reside na possibilidade efetuar a aquisição dos dados com períodos de tempo muito curtos (na ordem do segundo).

1.3 Objetivos

Para ultrapassar os inconvenientes referidos, pretendeu desenvolver-se uma solução que substitui e/ou trabalha em paralelo, com o sistema de monitorização já existente baseado na tecnologia cablada. Com a nova solução devem ser melhorados vários aspetos face ao sistema existente, nomeadamente a funcionalidade e interatividade, através do uso de novas tecnologias de hardware e software. Esta solução deverá ter em conta alguns requisitos, nomeadamente:

- Usar sistemas de aquisição e tratamento dos dados por hardware e software de baixo custo e genéricas, em detrimento de soluções de hardware mais complexas e dedicadas;

- Garantir a transmissão de dados sem fio com a mesma fiabilidade da transmissão cablada;

- Permitir a análise em tempo real dos dados recolhidos no momento e torná-la mais *user frendly*;

- Permitir o armazenamento dos dados adquiridos e ter acesso remoto a uma grande quantidade de informação.

Estes requisitos devem ser enquadrados ainda pela necessidade de o sistema permitir um tempo de aquisição o mais reduzido possível, na ordem de alguns segundos, em consonância com as constantes de tempo que este tipo de sistema de energia renovável possui.

1.4 Estrutura do relatório

Para o desenvolvimento do trabalho proposto foram seguidas várias etapas, descritas neste relatório. Na sequência do enquadramento do problema já apresentado, no próximo capítulo são apresentadas as soluções tecnológicas atuais, passíveis de ser

usadas no trabalho, assim como sistemas e problemas similares já analisados noutros estudos.

No capítulo 3 é descrita a arquitetura utilizada no desenvolvimento do sistema e justificado, consoante o ambiente, características em que o sistema será aplicado e resultado final pretendido, as escolhas feitas para o desenvolvimento do sistema.

No capítulo 4 é descrito o desenvolvimento e implementação de duas soluções para a aquisição e tratamento de dados, uma utilizando a Google Cloud e outra utilizando o servidor web.

O capítulo 5 apresenta os testes e os seus respetivos resultados e o capítulo 6 apresenta a conclusão final.

Capítulo 2 – Estado da arte

A monitorização e controlo de sistemas físicos é uma necessidade painel nas mais variadas áreas, podendo envolver a leitura de vários tipos de variáveis: elétricas, térmicas, climatéricas, hidráulicas, nucleares ou biológicas entre outras. No caso do PVT as quatro primeiras estão envolvidas. Tradicionalmente a tecnologia cablada foi a solução usada para a transmissão de informação em sistemas de instrumentação e medida. Contudo, quando a distância de transmissão é superior a alguns metros, pode haver perda de informação, sobretudo no caso da informação se apresentar na forma de tensão analógica. Como alternativas foram desenvolvidas soluções como a malha de corrente 4-20mA e a transmissão digital série como as RS 232 e 485. Contudo estas mantiveram a base cablada inicial. É sabido que a informação pode ser transmitida por ondas hertzianas, mas as tecnologias de base analógica, como a FM, não se revelaram alternativas suficientemente fiáveis. Com o desenvolvimento das tecnologias eletrónica e de informação digital ligadas à computação foram desenvolvidas novas formas de transmissão sem fios (wireless), nomeadamente o Wi-Fi. Esta tornou-se standard na ligação em rede de computadores no âmbito da Internet, mas com o aparecimento da designada Internet das Coisas (e não apenas computadores), outras formas de transmissão *wireless*, como o Bluetooth e o ZigBee e a computação e processamento na rede (*Cloud Computing*) é natural que se tire partido destas tecnologias no âmbito de aplicações como os sistemas de medida envolvendo energias renováveis.

Além da evolução nos sistemas de transmissão, a tecnologia eletrónica permitiu a integração em circuitos integrados (ou *chips*) de diversos componentes e funções necessárias à medida e processamento de variáveis físicas. Assim, componentes associados a funções como amostragem, multiplexagem, conversão analógico-digital e digital-analógica e microprocessadores integraram dispositivos conhecidos por microcontroladores, formando sistemas de hardware e software embebidos que facilitam os processos de aquisição de dados e atuação. Entre esses sistemas está a plataforma Arduino.

2.1 Arduino

Arduino foi inventado em 2005 por Massimo Benzi (Basconcello) com a finalidade de desenvolver uma placa de prototipagem eletronica de baixo custo para os estudantes do Instituto IVREA das areas de computação e eletrotecnia, visto naquela altura as placas de prototipagem eletronica serem ainda bastante caras.

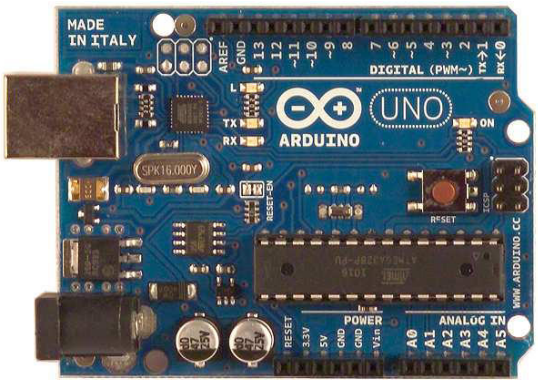


Figura 3 - Arduíno Uno (Basconcello)

O projeto foi criado para que o utilizador não tivesse de ter grandes conhecimentos em eletrónica ou programação para desenvolver os seus projetos. Foi originalmente baseado num microcontrolador Atmel AVR de 8bits com suporte a entradas e saídas analógicas e digitais embutidas, além de uma interface serial ou USB. A alimentação de energia pode ser feita por USB ou por uma fonte de alimentação externa. Outra importante característica do sistema foi ser *Open Source* permitindo que outras pessoas pudessem desenvolver novos componentes para integrar a plataforma Arduíno.

Com o passar do tempo foram desenvolvidas vários modelos Arduino de forma serem mais facilmente adaptados a varios tipos de projetos e necessidades. A Tabela 5 apresenta os modelos existentes atualmente.

Tabela 5 - Modelos de Arduíno existentes na altura da elaboração deste projeto (Arduino 1)

A programação do Arduino é feita através da comunicação serial. Para programar o Arduíno foi utilizada a interface gráfica Arduino IDE (Figura 4) multiplataforma construída em java mantida pela GitHub. A programação utilizada no Arduino tem origem no Wiring que é baseada em C/C++.

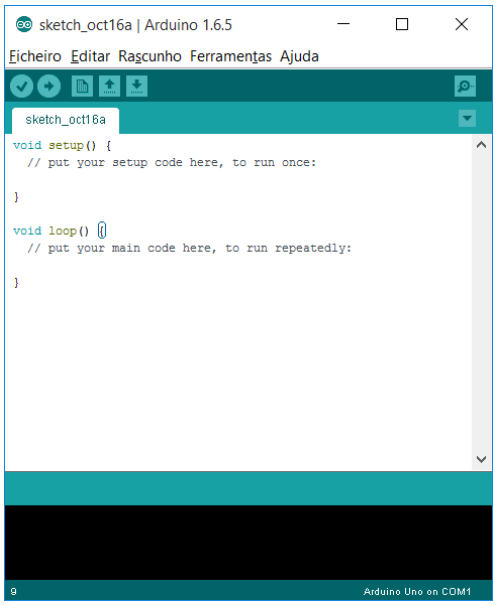


Figura 4 - Arduíno IDE

2.2 Computação móvel

As novas tecnologias de comunicação estão na origem da computação móvel que permite a utilização de dispositivos móveis portáteis para aceder a dados e serviços remotos via comunicação sem fios,"*anytime anywhere*". Esta evolução tecnológica só foi possível devido à evolução paralela de protocolos de comunicações, que garantem a qualidade de funcionamento dos sistemas em diferentes condições. A Tabela 6 apresenta as normas utilizadas nos sistemas de comunicação móveis baseados nos conhecidos telemóveis.

Tabela 6 - Normas utilizadas nos sistemas de comunicação móveis (Kumar)

Symbol	Standard	Full Name	Maximum Download Speed (Theoretical)	Maximum Upload Speed (Theoretical)
2G	GSM	Global System for Mobile Communications	14.4 Kbits/s	14.4 Kbits/s
G	GPRS	General Packet Radio Service	53.6 Kbits/s	26.8 Kbits/s
E	EDGE	Enhanced Data rates for GSM Evolution	217.6 Kbits/s	108.8 Kbits/s
3G	UMTS	Universal Mobile Telecommunications System	384 Kbits/s	128 Kbits/s
H	HSPA	High-Speed Packet Access	7.2 Mbits/s	3.6 Mbits/s
H+	HSPA+	Evolved High-Speed Packet Access - Release 6	14.4 Mbits/s	5.76 Mbits/s
H+	HSPA+	Evolved High-Speed Packet Access - Release 7	21.1 Mbits/s or 28.0 Mbits/s	11.5 Mbits/s
H+	HSPA+	Evolved High-Speed Packet Access - Release 8	42.2 Mbits/s	11.5 Mbits/s
H+	HSPA+	Evolved High-Speed Packet Access - Release 9	84.4 Mbits/s	11.5 Mbits/s
H+	HSPA+	Evolved High-Speed Packet Access - Release 10	168.8 Mbits/s	23.0 Mbits/s
4G	LTE	Long Term Evolution	100 Mbits/s	50 Mbits/s
4G	LTE-A	Long Term Evolution - Advanced	1 Gbits/s	500 Mbits/s

Os dispositivos móveis têm evoluído também em tamanho, reduzindo as suas dimensões e peso, mas mantendo a autonomia energética e aumentando as capacidades de processamento e interligação. Dependendo de características construtivas os dipositivos móveis podem classificar-se como *smartphones, tablets* ou *notebooks* e mais recentemente em *wearables*.

Figura 5 - Dispositivos móveis (WK3)

Os dispositivos móveis permitem aos seus utilizadores a execução de tarefas em movimento, em qualquer momento e lugar e com tempos de execução relativamente curtos (Figura 6).

Figura 6 - Características de dispositivos móveis (Nadav Savio, 2007)

Para tirar partido desta tecnologia é necessário software adequado para ser executado em dispositivos móveis variados, pelo que o seu desenvolvimento tem em foco especialmente a adaptação visual a esses dispositivos e o funcionamento com as características físicas de cada dispositivo.

Com a evolução da computação móvel surgiram novos conceitos como a internet das coisas e a computação na nuvem.

A internet das coisas, (Figura 7), é um conceito que tem como objetivo criar uma rede inteligente de objetos comuns, juntamente com serviços web, com a finalidade de partilharem informação para que esta possa ser utilizada nas mais diversas funções. Parte da infra-estrutura da internet das coisas é definida por tecnologias baseadas em sensores e Wi-Fi, já estabelecidas e noutras mais recentes como a RFID e NFC.

Figura 7 - Internet das coisas (123RF)

A computação na nuvem Figura 8), é um conceito que permite visualizar, alterar e armazenar documentos e aplicações do utilizador, fora do seu dispositivo de forma a poderem ser acedidos indiferentemente do dispositivo utilizado e da localização do utilizador, bastando para isso ter uma ligação ao serviço *cloud*. O fornecedor do serviço fica encarregue de tarefas como a manutenção, atualização ou *backup*.

Figura 8 - Computação na nuvem (Alecrim, 2008)

Existem atualmente vários serviços de *cloud* disponíveis, nomeadamente:

- Google Drive (Google Drive): é um serviço de armazenamento e sincronização de arquivos, apresentado pela Google em 24 de abril de 2012. Incorpora o Google Docs, um leque de aplicações de produtividade, que oferece a edição de documentos, folhas de cálculo, apresentações, e muito mais.

- OneDrive (OneDrive): é o serviço de armazenamento em nuvem da Microsoft com 7 GB grátis e com a possibilidade de adquirir mais espaço. Tem serviços sincronizados com o windows 8/10, windows phone e Xbox.

- iCloud (iCloud): sistema lançado pela Apple em 2011, é capaz de armazenar gratuitamente até 5 GB de fotos, músicas, documentos, livros e contactos, com a possibilidade de adquirir mais espaço em disco (pago).

- Dropbox (Dropbox): é um sistema de armazenamento em nuvem que se inicia gratuitamente com 2GB podendo, com a indicação de amigos, aumentar o espaço para armazenamento de arquivos até 18GB. Também tem opções pagas com maior espaço.

Estes conceitos e tecnologias são cada vez mais usados em sistemas de monitorização remota e servirão também de base ao projeto em estudo.

2.3 Exemplos de sistemas de monitorização

A aplicação das tecnologias descritas em sistemas de monitorização de energias renováveis e similares pode ser evidenciada nalguns exemplos:

Christopher McCoy (McCoy, 2011) desenvolveu um sistema denominado “Remote Weather Station Using XBee Wireless Transceivers”. O sistema passa pela utilização de um sistema microcontrolador da família Arduino para aquisição dos dados de uma estação meteorológica. Os dados são transmitidos do Arduino para um computador através do sistema de comunicação ZigBee.

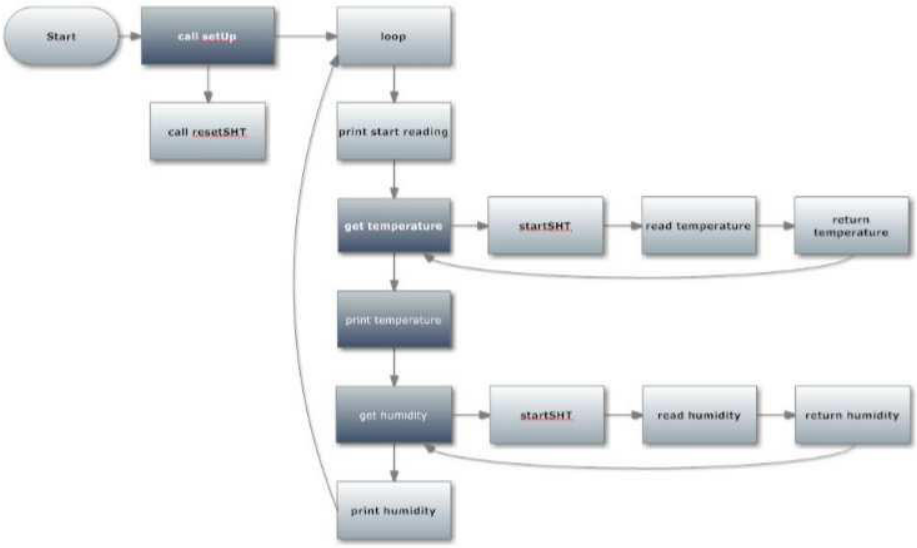


Figura 9 - Fluxograma do modelo utilizado no projeto "Remote Weather Station Using XBee Wireless Transceivers".

Esta solução resolve o problema da utilização de cabos para o envio de dados, no entanto o acesso continua a ter que ser feito no computador local.

Oliver Schwarts (Schwartz, 2015) usou um sistema denominado "Cloud-Connected Weather Station with the Arduino Yun & Temboo". Neste sistema (Figura 10) é utilizado também um Arduino para a leitura dos dados dos sensores que medem as variáveis meteorológicas. O Arduino usado é o Yun que vem com um módulo Wi-Fi integrado que lhe permite conectar a uma rede interna e assim disponibilizar os dados remotamente.

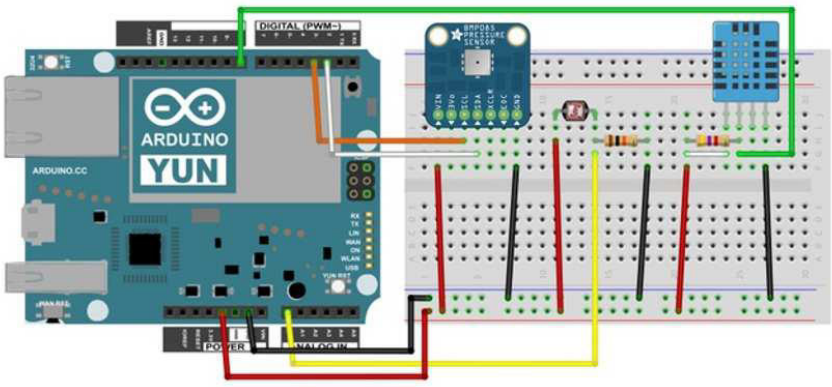


Figura 10 - Esquema desenvolvido por Oliver Schwarts

Os dados são enviados para um serviço c*loud*, neste caso o Google Drive, e tratados em *spreadsheets* (Figura 11). Dentro da *spreadsheet* é possível criar gráficos com os dados obtidos facilitando a análise destes pelo utilizador.

	A	B	C	D	E	F
1	**Time**	**Humidity**	**Light level**	**Pressure**	**Temperature**	**Altitude**
2	03/11/14-08:47:56	34	653	1002.54	23.77	92.43

Figura 11 - Spreadsheet dos dados adquiridos pelo sistema desenvolvido por Oliver Schwarts

O método utilizado para enviar os dados para o Google Drive foi através da biblioteca TemBoo que já vem pré instalada no Arduino Yun. A utilização deste método tem como limitação, na versão gratuita, só permitir o envio de 250 chamadas por mês, podendo tornar-se demasiado caro quando se pretende enviar uma grande quantidade de mensagens. Nesta solução o período de aquisição de dados é de 10 minutos.

S. Zahurula, e outros, (S. Zahurula, 2014) apresentaram em "Development of a prototype for remote current measurements of PV panel using WSN" uma solução baseada no Arduino e na tecnologia ZigBee para recolha dos dados e no programa LabView para o seu processamento (Figura 12).

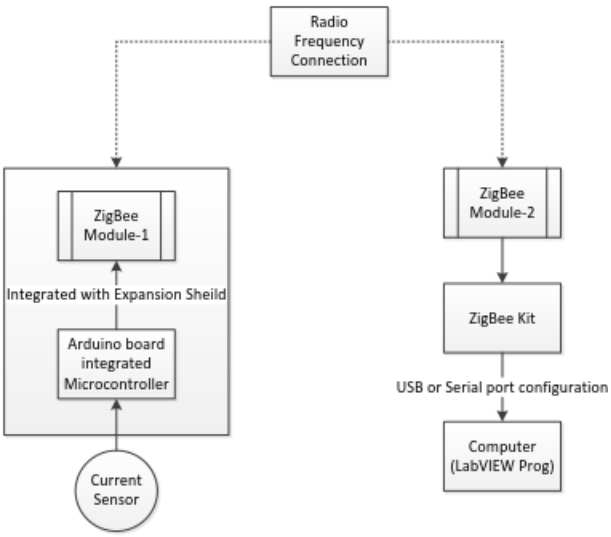


Figura 12 - Fluxograma do sistema desenvolvido por S. Zahurula, e outros (S. Zahurula, 2014)

Alguns fabricantes de dispositivos ligados às energias renováveis apresentam também soluções proprietárias para a monitorização dos seus equipamentos. Por exemplo a empresa SMA propõe a SUNNY WEBOX (SMA Solar Technology), uma solução que pode basear-se em tecnologia cablada série 485 ou sem fios e usa um portal WEB próprio para armazenamento dos dados.

Figura 13 - Sistema de monitorização da empresa SMA (SMA Solar Technology)

Com a análise dos exemplos apresentados foram ilustradas várias soluções que podem servir de base ao sistema que se pretende desenvolver. Assim verifica-se ser comum a solução baseada em plataformas microcontroladoras Arduino e na transmissão de dados, sem recurso à utilização de cabos, usando a tecnologia ZigBee. Por outro lado usam-se soluções baseadas na *cloud* que disponibilizam remotamente os valores dos sensores, recorrendo a ferramentas de criação de gráficos, de forma a facilitar a análise dos dados. Assim, o sistema que se pretende desenvolver neste projeto deverá utilizar soluções similares, permitindo o envio de muitos dados com o menor tempo de aquisição e custo possíveis.

Capitulo 3 – Arquitetura do sistema

Para conceção de uma solução para o problema concreto em estudo é necessário analisar as suas características físicas e técnicas. É importante também descrever o ambiente onde o sistema será instalado, para melhor compreensão das opções que serão tomadas em relação a determinados componentes em relação a outros.

3.1 Características e ambiente de trabalho do sistema

O sistema PVT está localizado numa área sem acesso a uma ligação de internet direta, mas a cerca 15m encontra-se o laboratório de máquinas elétricas da ESTG onde já existe uma ligação de sinal de internet por Wi-Fi (Figura 14). É para este laboratório que é encaminhada a informação por via cablada, assim como a energia elétrica produzida pelo sistema, pelo que este laboratório será utilizado para o desenvolvimento da nova solução a propor. No local onde está instalado o PVT existe um quadro elétrico com acesso à rede elétrica do IPG, permitindo a ligação de fontes de alimentação para a obtenção da energia para os sistemas eletrónicos a utilizar.

Figura 14 - Localização do sistema PVT

A análise ao funcionamento do sistema PVT requer a leitura de 12 variáveis físicas distintas – caudal do fluido, velocidade do vento, direcção do vento, temperatura ambiente, radiação solar, tensão do painel fotovoltaico, corrente do painel fotovoltaico, temperatura do vidro superior do painel fotovoltaico, temperatura inferior do vidro do painel fotovoltaico, a temperatura de entrada do fluido no PVT, a temperatura de saída do fluido do PVT e a temperatura posterior do PVT. Por natureza as leituras são do tipo analógico, mas podem depender dos sensores em causa. A Figura 15 mostra o esquema de posicionamento dos sensores e as ligações elétricas e hidráulicas do sistema

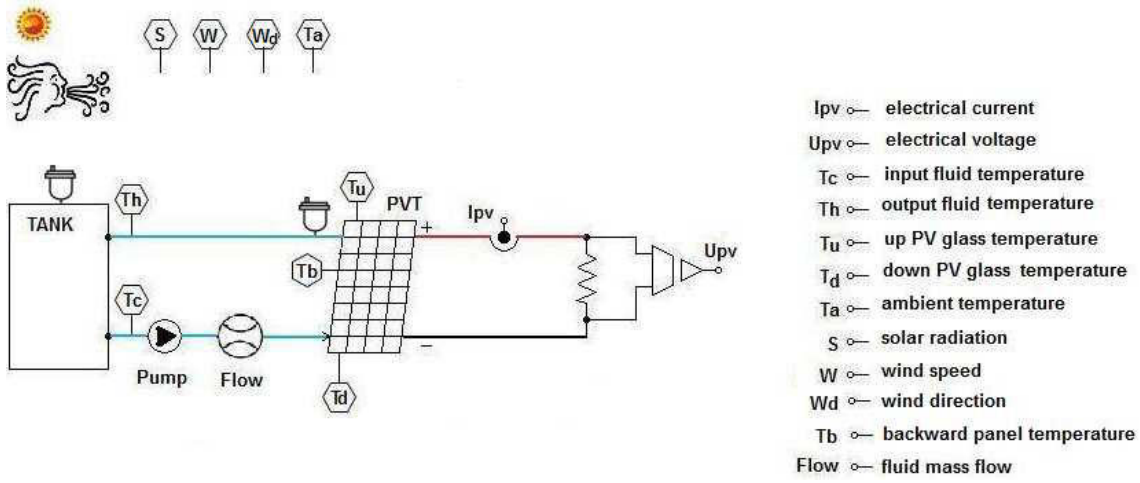


Figura 15 - Esquema de posicionamento dos sensores e as ligações elétricas e hidráulicas do sistema.

Os sensores estão ligados a circuitos de condicionamento de sinal, já usados pelo sistema de transmissão de dados cablado e que se pretendem manter ou adaptar para a transmissão *wireless* (circuitos no Anexo A1).

Na estação meteorológica, o sensor de radiação é um piranómetro com uma sensibilidade de 16.17 $\mu V/W/m^2$ (Kipp & Zonen). O valor convertido pelo sensor é amplificado até 5 V, de acordo com a radiação máxima admitida (1500 W/m^2).

A velocidade do vento é obtida através de um circuito resistivo com comutação magnética, que fornece uma informação em frequência com uma sensibilidade de 2.5 hz/ms^{-1} (Thies Clima). Esta frequência é convertida numa tensão analógica até 5 V,

usando o chip LM2917 (Texas Instruments), tendo em conta uma velocidade máxima de 120 km/h.

A direção do vento é fornecida de forma direta por um sensor resistivo potenciométrico, alimentado a 5V.

O sensor de temperatura ambiente é baseado num termistor de 10 kΩ. Este tipo de sensor encontra-se inserido num divisor resistivo e é também amplificado tendo em conta os valores admitidos para a temperatura (-20 a 50ºC).

O caudal do fluido hidráulico é medido através de um sensor de turbina, com pás magnéticas e usando o efeito Hall, fornecendo também uma informação em frequência com uma sensibilidade de (115 Hz/l/min). Esta frequência é convertida numa tensão analógica até 5 V, usando o chip LM2917, tendo em conta um caudal máximo de 5 l/s.

As temperaturas associadas ao vidro do painel e ao fluido são medidas com sensores resistivos PT100, inseridas num divisor resistivo e amplificadas até 5 V tendo em conta temperaturas mínimas máximas de -20 e 50 ºC.

O sistema cablado existente, já testado, servirá como referência para os dados obtidos com a nova forma de aquisição e transmissão. Quando necessário usar-se-ão também sistemas de medida portáteis autónomos, como outros termómetros, anemómetros, piranómetros e voltímetros, como o do Anexo A2 (Chauvin-Arnoux).

3.2 Comunicação de dados

Entre os principais objetivos a atingir no projeto está a transmissão *wireless* dos dados de forma a ser possível monitorizar remotamente o sistema e armazenar informação sobre o funcionamento do mesmo na *cloud*. Para tal é imperativo ter acesso à internet e, como foi referido no capítulo anterior, no local onde se encontra o PVT esta condição não se verifica. Uma solução imediata e direta passaria por usar uma transmissão GSM/GPRS, mas esta tem custos que se pretendem minimizar. Desta forma há a necessidade de enviar os dados do PVT para o laboratório com acesso à internet. Para esse efeito são necessários dois Arduinos, um instalado junto ao PVT, que reencaminha os dados para outro Arduino localizado no laboratório. Existem diferentes

soluções disponíveis para a comunicação entre Arduinos sendo as suas características apresentadas na Tabela 7.

Tabela 7 - Protocolos de comunicação sem fios (Churchill, 2007)

	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice/Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	.1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

A escolha recaiu sobre o ZigBee por ser a que melhor se adequa às necessidades do projeto, nomeadamente em termos alcance do sinal, baixo preço e baixo consumo de energia de cada módulo. Por isso esta tecnologia tem aplicações crescentes em aplicações de monitorização e controlo.

Para a implementação deste sistema de comunicação foram adquiridos dois módulos XBee XB24-Z7WIT-004 com antena externa da empresa Digi International (DIGI 1) e duas Xbee *shield*. As *shield* são módulos que facilitam a ligação entre os Arduinos e acessórios de comunicação e ligação externas. A configuração de cada módulo XBee foi feita com recurso a uma placa de programação (Figura 16) e da aplicação gratuita XCTU disponibilizada pela Digi (Figura 17).

Figura 16 - Placa de programação de XBee (Oliveira, 2012)

Pretende-se que a comunicação seja do tipo ponto a ponto e que os dois módulos só comuniquem entre eles. Para esse efeito é necessário que tenham uma configuração idêntica. Na Tabela 8 podem ser vistas as configurações utilizadas nos dois módulos.

Tabela 8 - Configuração utilizada nos módulos XBee

Baud Rate	9600
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None

Na configuração dos módulos foi necessário definir o endereço de destino para cada módulo, através da configuração do DH (*destiny high address*) e o DL (*destiny low address*), o canal de comunicação e o PAN ID (nome da rede).

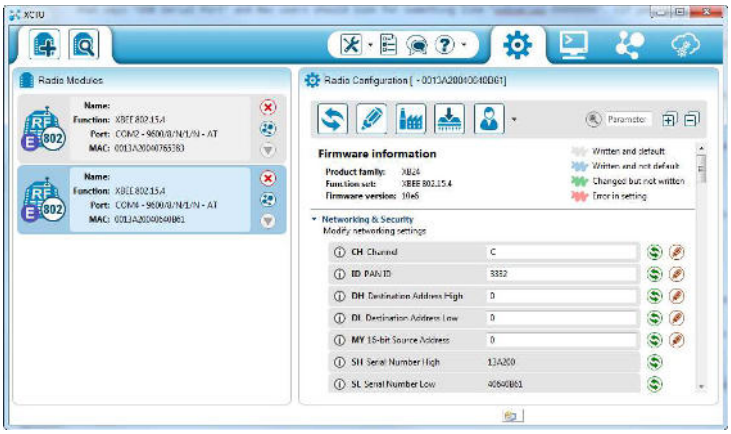


Figura 17 - Configuração de um módulo XBee utilizando a aplicação XCTU.

3.3 Aquisição de dados

Após selecionado o protocolo de comunicação entre os dois dispositivos Arduino foi necessário escolher as plataformas microcontroladoras entre os modelos de Arduino disponíveis. Nesta escolha é tida em consideração que cada um tem tarefas distintas bem especificadas.

Os requisitos principais para o Arduino que se encontra instalado junto ao PVT são os relacionados com a necessidade de ter pelo menos 12 entradas analógicas para suportar a conexão aos sensores existentes e suportar a *shield* para o XBeee. Foram analisados vários modelos de forma a encontrar o modelo que melhor se adequasse as tarefas pretendidas. A Tabela 9 apresenta as especificações dos modelos analisados mais relevantes.

Tabela 9 - Características de Arduinos (Arduino 1)

Arduino Model	UNO	Leonardo	MEGA	Yún
Microcontroller	ATmega328P	ATmega32u4	ATmega2560	ATmega32u4
Operating Voltage	5V	5V	5V	5V
Input Voltage (recommended)	7-12V	7-12V	7-12V	5V
Input Voltage (limits)	6-20V	6-20V	6-20V	
Digital I/O Pins	14 (6 PWM)	20	54 (15 PWM T)	20
PWM Channels	6	7	15	7
Analog Input Channels	6	12	16	12
DC Current per I/O Pin	20 mA	40 mA	40 mA	40 mA
DC Current for 3.3V Pin	50 mA	50 mA	50 mA	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader	32 KB (ATmega32u4) of which 4 KB used by bootloader	256 KB (of which 8 KB used by bootloader)	32 KB (ATmega32u4 of which 4 KB used by bootloader)
SRAM	2 KB (ATmega328P)	2.5 KB (ATmega32u4)	8 KB	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega328P)	1 KB (ATmega32u4)	4 KB	1 KB (ATmega32u4)
Clock Speed	16 MHz	16 MHz	16 MHz	16 MHz

O Arduino Uno só tem 6 portas analógicas o que exclui o seu uso. O Arduino Leonardo cumpre os requisitos exigidos e é mais barato que os restantes modelos, apresentando-se assim como a melhor solução para este sistema. Possui também ligações digitais para acções externas de ligar/desligar a bomba que faz circular o fluido e de controlar a corrente do painel fotovoltaico.

No caso do Arduino que se encontra dentro do laboratório, além de a *shield* para o XBeee, o principal requisito é o de permitir a comunicação via Wi-Fi. Para a comunicação Wi-Fi existe a possibilidade de esta já estar integrada no Arduino (caso do Yun) ou ter que se investir num módulo externo. O Arduino Yun é uma plataforma que junta o ambiente Arduino com o ambiente Linux (conhecido por Linino) e possui ainda um leitor de cartões SD. O módulo Wi-Fi e o leitor de cartões SD são tratados do lado do Linux (Figura 18), o que simplifica o manuseamento daqueles em relação a uma plataforma Arduino convencional com *shields*.

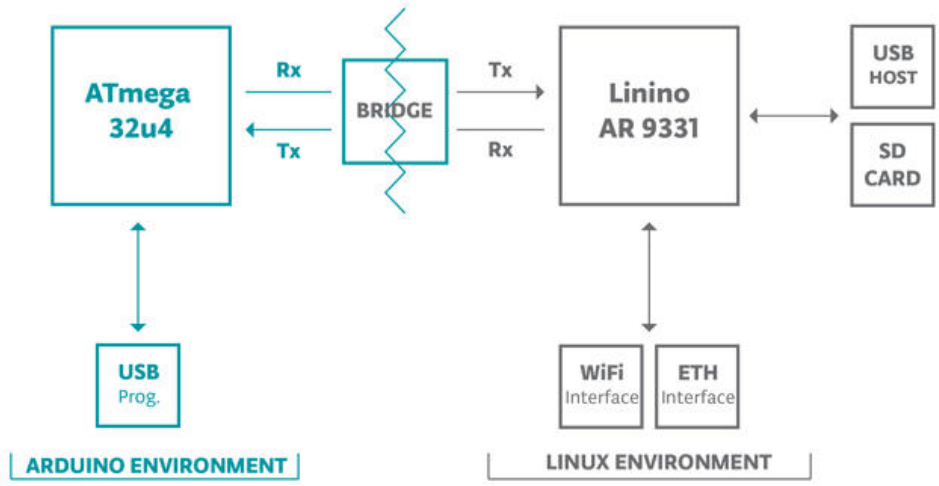


Figura 18 - Arquitetura do Arduino Yun (Arduino 2)

Além da facilidade de trabalhar com o Wi-Fi e cartão SD, o ambiente Linux dá acesso à data do sistema, o que é extremamente útil para um sistema de energia renovável e traz pré-instalado um servidor web. Analisadas as características do Arduino Yun, optou-se por esta plataforma devido a corresponder todos os requisitos do sistema e ainda trazer funcionalidades adicionais que podiam ser exploradas no projeto.

Tabela 10 - Características do Arduino Yun (Linux) (Arduino 2)

Processor	Atheros AR9331
Architecture	MIPS @400MHz
Operating Voltage	3.3V
Ethernet	IEEE 802.3 10/100Mbit/s
Wi-Fi	IEEE 802.11b/g/n
USB Type-A	2.0 Host
Card Reader	Micro-SD only
RAM	64 MB DDR2
Flash Memory	16 MB

Concluída a definição do sistema de monitorização obteve-se o esquema apresentado na Figura 19.

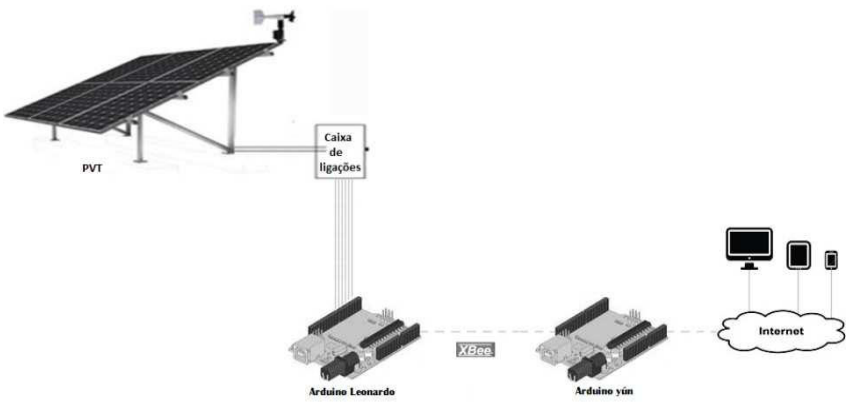


Figura 19 - Sistema final de monitorização PVT

O esquema anterior ilustra a estrutura de hardware da solução concebida para a monitorização remota do PVT. Outra componente da solução está associada às aplicações de software relacionadas com as plataformas Arduino e com a *cloud*.

Capitulo 4 – Desenvolvimento e implementação da solução

No desenvolvimento da solução idealizada, foram primeiro desenvolvidas as aplicações para as plataformas Leonardo e Yun, garantindo a correta aquisição dos valores dos sensores e a comunicação entre as duas plataformas via Xbee. Só posteriormente foi introduzida a ligação Wi-Fi com ligação à internet e computação na *cloud*.

4.1 Aplicação Leonardo

A Figura 20 mostra os componentes de hardware reais usados para a aplicação do Arduino Leonardo. Além do Arduino Leonardo e módulo Xbee e respetiva *shield*, referidos no capítulo anterior, foi usada uma *screw shield* para facilitar as ligações elétricas dos sensores (Anexo A3).

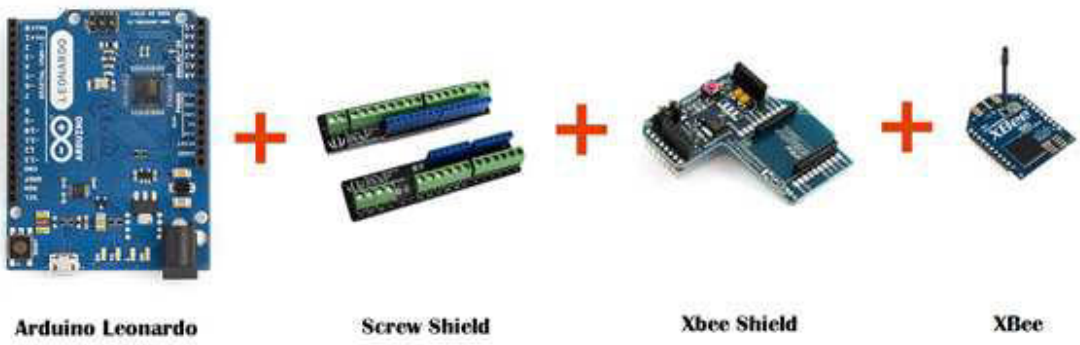


Figura 20 - Estrutura de hardware física para a aplicação Leonardo

Ao Arduino Leonardo vão estar ligados os 12 sensores de monitorização, através dos seus circuitos de condicionamento de sinal. Na Tabela 11 é apresentada a relação das portas analógicas do Arduino Leonardo com os sensores.

Tabela 11 - Ligações analógicas

Entrada Analógica	Sensor	Variável
0	Caudal	Caudal (Flow)
1	Velocidade do vento	WindSpeed
2	Direção do vento	WindDir
3	Temperatura ambiente	AmbTemp
4	Radiação solar	SunRad
5	Tensão painel	PVoltage
6	Corrente painel	PVCurrent
7	Temperatura superior do vidro	UpPVTemp
8	Temperatura inferior do vidro	DownPVTemp
9	Temperatura entrada do fluido	InFluidTemp
10	Temperatura saída do fluido	OutFluidTemp
11	Temperatura traseira do painel	BackPVTemp

Tendo em conta as suas funções, a aplicação do Arduino Leonardo foi concebida de acordo com o fluxograma da Figura 22. Analisando o fluxograma, pode verificar-se que o Arduino Leonardo vai estar indeterminadamente à espera de um pedido para ser feito a leitura dos sensores e posterior envio dos mesmos. O pedido é feito através da receção de uma mensagem com as seguintes características (Figura 21):

*	SEND	#

Figura 21 - Mensagem de pedido de dados

Sendo:

* → caracter indicador de inicio de mensagem;

→ caracter indicador de fim de mensagem.

Esta mensagem é recebida via XBee e enviada unicamente e exclusivamente pelo Arduino Yun. Assim que for identificado um pedido de dados, o Arduino Leonardo vai ler os sensores do sistema e efetuar o tratamento matemático dos mesmos.

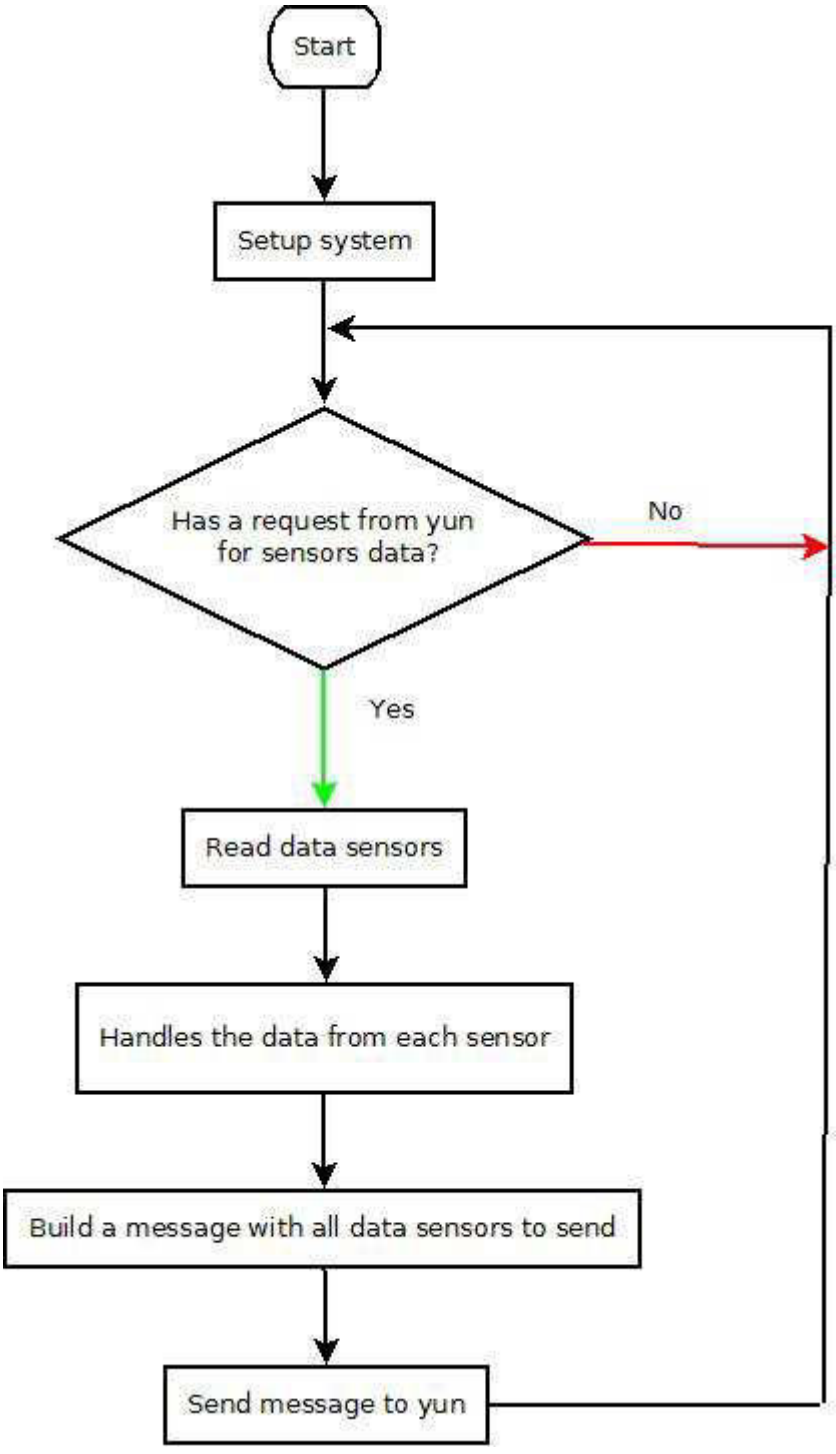


Figura 22 - Fluxograma da aplicação do Arduino Leonardo

Com vista a minimizar erros de leitura foi prevista a possibilidade de cada sensor ser lido mais que uma vez (3 por defeito) calculando-se depois a média das leituras. Esta possibilidade está ilustrada no código seguinte.

```
//lê todos os sensores x vezes (x = 'numeroLeituras' ) ,calcula a média e guarda os valores na array
//sensorValues'
void receiveVal(){
 for(int b=0;b<numeroLeituras;b++){  //numero de leituras pretendidas
  for(int i=0;i<tamanhoArray;i++){       // lê os 12 sensores
      sensorValues [i] +=(float)analogRead(myPins[i]); // lê o valor do sensor e adiciona-o ao valor anterior
  }
 }
 for(int i=0;i<tamanhoArray;i++){
  sensorValues [i]=sensorValues [i]/numeroLeituras;  //calcula a média para cada sensor
 }
}
```

O tratamento dos valores de cada sensor é feito tendo em conta a curva de calibração de cada um (já existente na aplicação MatLab e reconfirmada), a que corresponde uma fórmula matemática de acordo com a Tabela 12.

Tabela 12 - Formulas para tratar os dados adquiridos de cada sensor

Sensor	**Fórmula** (SV = Valor sensor; k1= SV *5.0/1023.0)
Flow	SV *5.0/1023.0*0.03/4.80
WindSpeed	SV *5.0/1023.0*30.0/4.80
WindDir	SV *5.0/1023.0*360.0/5.0
AmbTemp	k1^6 * 0.2715 - k1^5 * 4.1324 + k1^4 * 23.641 -k1^3 * 61.447 + k1^2 * 66.787 + k1 * 3.4292 - 52.64
SunRad	SV *5.0/1023.0*1250.0/4.8
PVoltage	SV *5.0/1023.0*50.0/4.8
PVCurrent	SV *5.0/1023.0*6.0/4.80
UpPVTemp	k1^2 * 0.7219 + k1 * 16.861 - 27.786
DownPVTemp	k1^2 * 0.7219 + k1 * 16.861 - 27.786
InFluidTemp	k1^2 * 0.7219 + k1 * 16.861 - 27.786
OutFluidTemp	k1^2 * 0.7219 + k1 * 16.861 - 27.786
BackPVTemp	k1^2 * 0.7219 + k1 * 16.861 - 27.786

Após os dados de cada sensor estarem tratados, é construída a mensagem a ser enviada para o Arduino Yun. Para construir a mensagem foi desenvolvido um protocolo de segurança, que vai permitir a deteção do início e fim da mensagem e utiliza uma forma simples de detetar falhas na mensagem. Para a deteção de erros na mensagem, é feita a contagem dos caracteres de todos os valores dos sensores, inclusive dos caracteres de separação (neste caso a ‘,’). A estrutura da mensagem enviada assume assim a forma mostrada na Figura 23.

*	Tamanho da mensagem	Valores dos sensores	#

Figura 23 - Estrutura da mensagem enviada com os dados dos sensores

Um exemplo de mensagem será: *51,0.0001,4.7,197,7.4,0,1,0.14,7.2,7.2,4.8,13.8,12.9,#

Assim que a mensagem estiver construída é enviada via XBee para o Arduino Yun, com a função de envio de mensagem ilustrada no código seguinte.

```
void sendData(){
 //formato da string a enviar
  //1 caracter, '#' indica inicio da mensagem
  //2 ..., caracteres com o tamanho da mensagem
  //3 ...* caracteres, valores dos 12 sensores separados por virgulas
  //ultimo caracter, '*' sinaliza o final da mensagem

      tamanhoMsg=mensagem.length();
            Serial1.print('#');
      Serial1.flush();
            Serial1.print(tamanhoMsg);
      Serial1.flush();
            Serial1.print(',');
      Serial1.flush();
      Serial1.print(mensagem);
      Serial1.flush();
            Serial1.print('*');
      Serial1.flush();
}
```

Após o envio da mensagem o sistema fica em espera de um novo pedido de dados.

4.2 Aplicação Yun

Para a aplicação do Yun os componentes de hardware reais usados estão ilustrados na Figura 24. Além do Arduino Yun, módulo Xbee com respetiva *shield* foi usado um botão de pressão. Este *push-button* irá ser usado para permitir interromper manualmente o sistema, anulando o pedido de novos dados e retomar os pedidos sempre que o utilizador pretender.

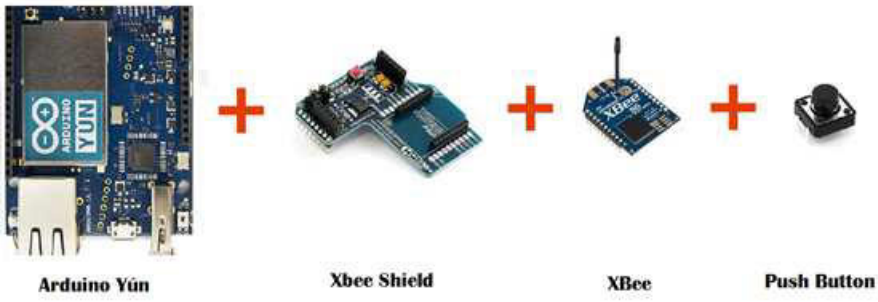


Figura 24 - Estrutura de hardware para a aplicação Yun

Na implementação inicial do *push button* encontrou-se um problema, pois cada vez que o botão era premido, eram detetados falsas ordens devido ao efeito de trepidação (*switch bouncing*), descrito na Figura 25:

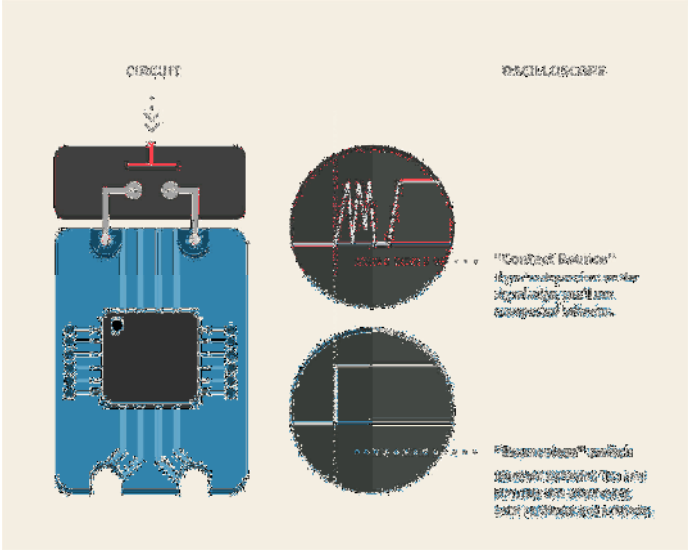


Figura 25 - Efeito switch bouncing (Clariá, 2012)

Este problema foi resolvido através dos métodos de interrupção existentes na família Arduino (Monk, 2013), usando as seguintes linhas de código (Arduino 3)

```
pinMode(2,INPUT_PULLUP);
attachInterrupt(2,turnOnOff,FALLING);
```

A primeira linha inicializa o pino 2 digital, como um entrada (*input*) com a resistência interna *pull-up* ativa. Assim, quando o botão não estiver pressionado a resistência de *pull-up* impõe a entrada a 5 V e o sinal obtido pelo Arduino é '1' ou 'High'. Quando o botão é pressionado o pino é ligado à 0 V (*gnd*) obtendo um sinal de '0' ou 'LOW'.

A segunda linha define a interrupção no Arduino. O primeiro parâmetro define o número da interrupção. O Arduino Yun (tal como o Leonardo) disponibiliza as interrupções ilustradas na Tabela 13:

Tabela 13 - Pinos usados pelas interrupções no Arduino Yun

Arduino	Pinos usados pelas interrupções
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7

Nesta aplicação foi seleccionada a interrupção associada ao pino 2. O segundo parâmetro define a função a ser chamada sempre que for acionada uma interrupção. O terceiro parâmetro foi definido como 'Falling' o que quer dizer que a interrupção é ativa sempre que o pino passar de 'High' para 'Low'. Por fim é colocada a função que é chamada sempre que existir a interrupção. O que esta função faz é mudar o estado do sistema e acionar o led do Arduino Yun para ligado/desligado conforme o estado deste. O código corresponde a esta função é o seguinte:

```
void turnOnOff(){
        estado= ! estado; // altera o estado do sistema
        //sinaliza o estado do sistema com o led 13 do arduino
         if(estado==HIGH){ digitalWrite(13, HIGH); }
        else { digitalWrite(13, LOW); }
}
```

Visto o Arduino Yun utilizar as portas 0 e 1 para fazer a comunicação entre o processador do Arduino propriamente dito e do Linino, foi preciso utilizar portas

digitais para criar uma ligação série virtual e fazer a comunicação entre o Arduino e o XBee. As portas virtuais foram criadas com recurso à biblioteca SoftwareSerial.h. A Tabela 14 mostra as portas digitais do Arduino utilizadas e a sua função.

Tabela 14 - Ligações digitais do Yun

Portas digitais	Função
2	*Push Button*
10	RX XBee
11	TX Xbee

No Arduino Yun não houve necessidade de usar portas analógicas. Por outro lado, além da comunicação Xbee o Yun vai também comunicar com a internet e pode interagir ainda com o cartão SD para armazenamento de dados. Assim, foi necessário definir três etapas de execução na aplicação do Arduino Yun, a saber:

- Requisição e tratamento de dados;
- Armazenamento de dados no cartão SD;
- Armazenamento de dados na nuvem.

A possibilidade de usar o cartão SD como meio de armazenamento, foi para garantir não se perderem dados no caso da ligação à internet ou do serviço *cloud*.

Para melhor compreensão do funcionamento do Yun, é apresentado na Figura 26 o seu fluxograma de execução de tarefas. O fluxograma ilustra desde já a utilização da plataforma da Google associada à *cloud*, cuja justificação será feita no capítulo seguinte, assim como a obtenção e processamento de informação associada ao tempo, variável normalemente necessária em sistemas envolvendo condições climatéricas.

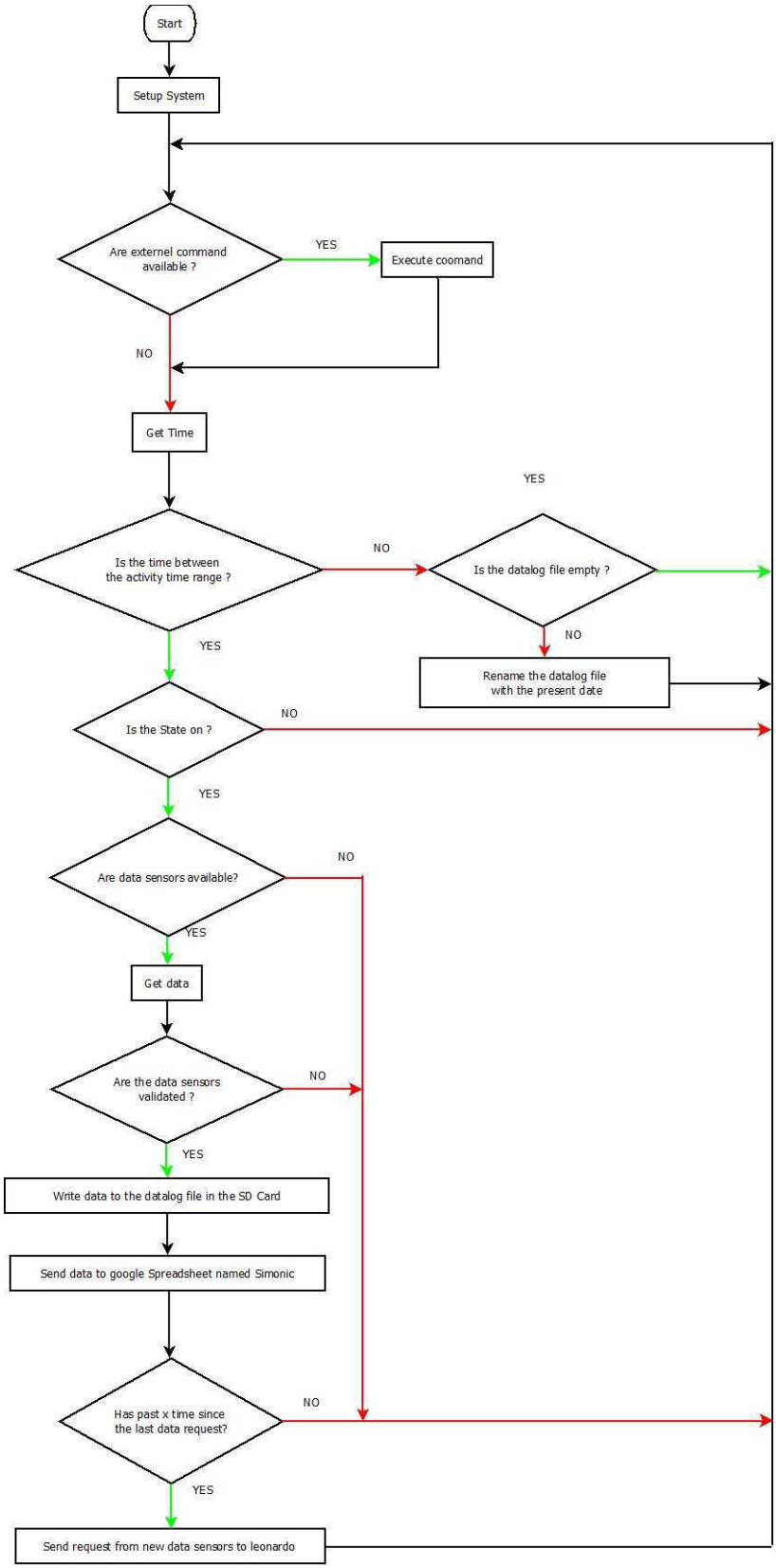


Figura 26 - Fluxograma do Arduino Yun.

4.2.1 Requisição e tratamento de dados

O objetivo do Arduino Yun é diariamente, dentro de um determinado período de tempo, requerer valores dos sensores ao Arduino Leonardo, armazenar os dados e disponibilizá-los remotamente.

O período de atividade do Arduino Yun é definido com base nas horas do dia em que existe maior luz solar, que variam ao longo do ano. Uma solução simples para esta definição seria, por exemplo, definir períodos diferentes para o inverno e para o verão. Foi esta a opção inicial testada. Posteriormente evoluiu-se para um sistema baseado na hora solar, que define períodos diferentes todos os dias, tendo em conta os algoritmos apresentados em (Eicker, 2003) a que corresponde o código seguinte:

```
int sunSetRise(int dayOfYear, int opcao){                    // opcao 1= sunrise opcao 2 = sunset
            float t = 0.0 , H = 0.0, pi = atan(1)*4, CT = 0, localOffset=0.0;
            //calc Latitude
            float lat = latitude * pi/180.0;
            //calc long
            float longi = longitude * pi/180.0;
            //calc constanta
            float BB = 360.0/365.0*(dayOfYear-1)*pi/180.0;
            //calc declinacao
            float dcl = 23.45*sin((360.0/365.0 * (284.0+dayOfYear))*pi/180.0 )*pi/180.0;
            //calc constante
            float Et = 229.2/60.0*(0.000075+0.001868*cos(BB)-0.032077*sin(BB)-0.014615*cos(2.0*BB)-
0.040849*sin(2.0*BB));
            //calc long correction
            float Lc = 4.0*(longi-0)/60.0;
            //calc sunrise hour angle
            float sun = -acos(-tan(lat)*tan(dcl))*180.0/pi;
            //calc true sunrise hour angle time
            float TLTsun  =sun/15.0+12.0;
            if (dayOfYear>89 && dayOfYear<209)
                        CT=TLTsun-Et+Lc+1.0; else
                       CT=TLTsun-Et+Lc;
             if(opcao==1)
                        return (int)CT; else{
             float Dh = 2.0/15.0*180.0/pi*acos(-tan(lat)*tan(dcl));
            CT=CT+Dh+1.0;
            return (int)CT;
    }
}
```

Como dados este código recebe recebe o dia do ano (entre 1 e 365/6), a latitude e longitude do lugar e retorna a hora de nascer do sol (CT) e as horas diárias de sol (Dh), que vão definir a hora de pôr-do-sol. A obtenção do dia é feita com recurso à data do sistema do Linino, Para a aquisição da data e/ou hora foi criada uma função getTimeStamp(), (Tabela 15).

Tabela 15 - Comandos suportados pela função getTimeStamp()

getTimeStamp	
Parâmetros de entrada	Retorno
a	Retorna *String* com data e hora atual no formato y/m/d-h:m:s
d	Retorna *String* com data atual no formato y/m/d
t	Retorna *String* com as horas atuais no formato h:m:s
h	Retorna *String* com a hora atual
D	Retorna *String* com o dia do ano atual

Durante o período de tempo definido, o Arduino Yun irá estar constantemente a pedir dados ao Arduino Leonardo, caso o sistema não tenha sido interrompido pelo utilizador. O pedido de dados é feito através da mensagem 'SEND' já referida na aplicação Leonardo. Assim que os dados do Arduino Leonardo são recebidos, são analisados e tratados para poderem ser armazenados no cartão SD ou disponibilizados remotamente. Para a análise da mensagem, o Arduino Yun vai considerar uma mensagem valida quando esta se encontrar entre os caracteres limitadores '#' e '*' (conforme detalhado no subcapítulo da aplicação Leonardo). Caso se verifique esta condição, será extraído o tamanho da mensagem e comparado com o tamanho da mensagem recebida. No caso de os tamanhos coincidirem a mensagem é considerada sem erros, caso contrário será ignorada. O código de execução é o seguinte:

```
//retira o tamanho da mensagem da mensagem recebida pelo leonardo
 indiceF=dataRc.indexOf(',');
 tamanho=dataRc.substring(indiceI,indiceF).toInt();
 dataRc=dataRc.substring(indiceF+1);
      // verifica se o tamanho da mensagem recebida é igual ao da mensagem enviada
      if(tamanho==dataRc.length() && tamanho>2){
        indiceI=0;
        indiceF=dataRc.indexOf(',');
       //guarda os valores dos sensores separadores num array, utilizado para enviar para a Google
        for(int i=0;i<numeroSensores;i++){
          vSensores[i]=dataRc.substring(indiceI,indiceF);
          indiceI=indiceF+1;
          indiceF=dataRc.indexOf(',',indiceI);
        }
```

Após a aceitação da mensagem sem erros o seu tratamento é feito por dois métodos distintos, consoante o seu fim, armazenar os dados no cartão SD ou na *cloud*:

- Tratamento para armazenamento no cartão SD

Aos valores dos sensores recebidos é adicionada a data e horas da receção e enviada para o cartão SD. Visto os dados serem datados no Arduino Yun e não no Leonardo que é o que faz a leitura dos sensores, vai existir uma margem de erro temporal aproximada de alguns milisegundos. A decisão de datar a leitura no Arduino Yun foi uma questão de diminuir os custos do sistema, visto que no Arduino Yun é possível aceder ao relógio interno deste. No Arduino Leonardo para se ter uma datação minimamente fiável seria necessário usar recursos externos a este.

- Tratamento para disponibilizar remotamente

Neste caso os dados de cada sensor são guardados individualmente num *array* com o tamanho do número de sensores esperados, neste caso 12 sensores até estes serem enviados para o serviço *cloud* escolhido. Após os dados serem armazenados, e caso ainda se esteja dentro do período de leitura, será enviado um novo pedido de dados ao Arduino Leonardo.

4.2.2 Armazenamento de dados no cartão SD

Os dados são enviados juntamente com a data para um ficheiro denominado datalog no formato .txt.

```
//envia para o cartao SD
        String aData=getTimeStamp('a');              // recebe a data no formato AAAA/MM/DD-hh:mm:ss
        aData+=',';                                  //adiciona uma virgula a seguir a data
        dataRc = aData+dataRc;                       //adiciona a data a mensagem
        File dataFile = FileSystem.open(openDatalog, FILE_APPEND);  // guarda os dados no ficheiro datalog
        if (dataFile) {
         dataFile.println(dataRc);                   //escreve no cartão SD
         dataFile.flush();
         dataFile.close();
          dataToSave=1;                              //aviso que existe dados a serem guardados
         }
         dataRc="";
```

Foi escolhido este formato para facilitar a leitura destes via programação e permitir no futuro, querendo, exportar os dados para uma base de dados. O ficheiro datalog.txt é criado assim que são enviados os primeiros dados diários para o cartão SD. No final de cada período diário o ficheiro datalog.txt é renomeado com a data do dia de receção (exemplo 20150221.txt) e enviado para uma pasta específica. O código associado à criação deste ficheiro é o seguinte:

```
void saveDay(){
   String newName =getTimeStamp('d');          //recebe data do dia no formato AAAAMMDD esta data vai ser
                                               //usada como nome do ficheiro com os dados do dia
  newName +=F(".txt");                         //define o formato do ficheiro
  Process p;
      p.runShellCommand(updatehistoric + newName);     //copia os dados do datalog.txt para o novo ficheiro
      p.runShellCommand(removeDatalog);                //apaga o ficheiro datalog
      p.flush();
      p.close();
}
```

Para evitar também a perda de dados por motivo de interrupção do sistema, seja por opção do utilizador, falha de energia ou outra, sempre que são enviados os primeiros dados do dia é verificado se já existe um ficheiro datalog no cartão de memória SD. Caso exista, significa que existem dados pendentes. Para evitar a perda desses dados, antes de serem guardados os novos dados é retirada a data do início da primeira linha do ficheiro e renomeado o ficheiro com essa data. No caso de no final do período diário já existir um ficheiro com essa data, os dados contidos no datalog serão adicionados no final do ficheiro já existente. No final o ficheiro datalog fica com o aspeto seguinte:

```
6/20/2015 10:38:14,0,1.3,105,28.5,707,25.9,3.9,44.2,44.1,25,43.3,28.3
6/20/2015 10:38:30,0,2.3,345,28.3,709,25.8,3.9,44,44.2,25.3,43.7,28.2
6/20/2015 10:38:47,0,0.2,357,28.4,706,25.6,3.9,44.9,44.7,25,43,27.7
6/20/2015 10:39:03,0,0,358,29.3,709,25.6,3.9,44.7,44.4,25.5,43.7,28
6/20/2015 10:39:19,0,2.7,3,29,712,25.9,3.9,44.7,44.2,25.4,44.2,28.4
6/20/2015 10:39:36,0,0.4,50,28.4,711,25.8,3.9,44.7,44.3,25.3,43.5,28.1
6/20/2015 10:39:51,0,1,102,28,713,26.1,3.9,45,44.1,24.8,43.1,28.3
6/20/2015 10:40:06,0,0.8,59,28.3,715,26,3.9,44.3,44.2,25.3,43.3,28.2
6/20/2015 10:40:22,0,0.7,58,28.4,715,25.9,3.9,44.4,44.4,25.5,43.5,28.1
6/20/2015 10:40:39,0,0.7,65,28.3,715,26,3.9,44.3,44.3,25.4,43.5,28.4
6/20/2015 10:40:55,0,2.1,45,27.7,720,26,3.9,43.3,44.1,25.7,43.8,28.6
6/20/2015 10:42:40,0,0.2,168,27.8,723,26.4,4,43.8,43.9,24.9,43.2,28.1
6/20/2015 10:42:54,0,0.5,16,28.4,723,26.3,4,43.7,44.3,25.5,43.9,28.1
6/20/2015 10:43:10,0,0.1,26,28.8,726,26.4,3.9,43.7,44.2,25.5,44.2,28.2
6/20/2015 10:43:27,0,1.1,83,28.7,728,26.5,4,43.8,44,25.4,44.5,28.6
6/20/2015 10:43:44,0,0.5,43,29,730,26.4,4,43.5,44.1,25.6,44.6,28.6
```

Figura 27 - Ficheiro datalog com os dados dos sensores do presente dia

4.2.3 Disponibilização dos dados remotamente

A disponibilização remota dos dados idealizada inicialmente pretendia ser baseada num dos serviços *cloud* do tipo do apresentado no capítulo anterior em 2.2, mas com a evolução do trabalho foi desenvolvida uma solução complementar baseada no servidor web do Arduino Yun.

4.2.3.1 Solução *cloud*

Para disponibilizar os dados remotamente aos utilizadores foram analisadas várias soluções e serviços. Essas soluções deveriam cumprir os requisitos já apresentados, tendo em consideração:

- Solução sem custos acrescidos;
- Permitir acesso remoto aos dados;
- Permitir visualizar dados em tempo real com opção gráfica;
- Tempo de disponibilização dos dados;

Foram comparadas várias soluções, destacando-se as que envolvem os seguintes serviços:

- Google drive;
- DropBox (com DataHero);
- Plotly.

O Google Drive (Google Drive) é um serviço *cloud* da Google que permite o armazenamento, sincronização e partilha de dados. O plano gratuito do Google Drive disponibiliza 15 GB de armazenamento e a possibilidade de trabalhar com algumas aplicações de produtividade. Em relação ao sistema que se pretende desenvolver este serviço oferece uma boa capacidade de armazenamento, facilidade de acesso e partilha dos dados e ferramentas para facilitar a sua análise, como folhas de cálculo com a possibilidade de apresentação de gráficos. O aspeto menos positivo é o tempo de envio dos dados. Um dos objetivos do sistema é que o período entre aquisição de dados seja o menor possível. Contudo o tempo requerido para enviar os dados para o serviço da Google aumenta consideravelmente com a quantidade de dados a enviar, não permitindo o seu conhecimento em tempo real e aumentando o tempo entre aquisição de dados. Para resolver este problema pensou-se numa forma de instalar o Google Drive no próprio sistema. Desse modo os dados poderiam ser gravados directamente numa pasta partilhada pelo Google Drive, passando esta a ser sincronizada autonomamente. Nesse caso o tempo de sincronização não iria influenciar o tempo de aquisição de dados. Visto não ser possível instalar a aplicação no Linino seria necessário obter dispositivos extras para essa função, o que iria aumentar substancialmente os custos do sistema. Ainda assim, conforme visto no capítulo 2, este serviço é usado em aplicações similares ao desenvolvido.

DropBox (Dropbox) é um serviço *cloud* que pertence ao Dropbox Inc. Este serviço, à semelhança do Google Drive permite o armazenamento, sincronização e

partilha de ficheiros. O plano gratuito oferece 2 GB de armazenamento. Em relação ao sistema que se pretende desenvolver, o espaço fornecido é muito inferior ao do Google Drive e não tem nenhuma ferramenta própria que facilite a receção e análise dos dados. Contudo este serviço pode ser associado com outros para ultrapassar estas questões.

O Temboo (Figura 28) é um serviço que fornece uma biblioteca que vem pré-instalada no Arduino Yun e que facilita a ligação deste com vários serviços *cloud*.

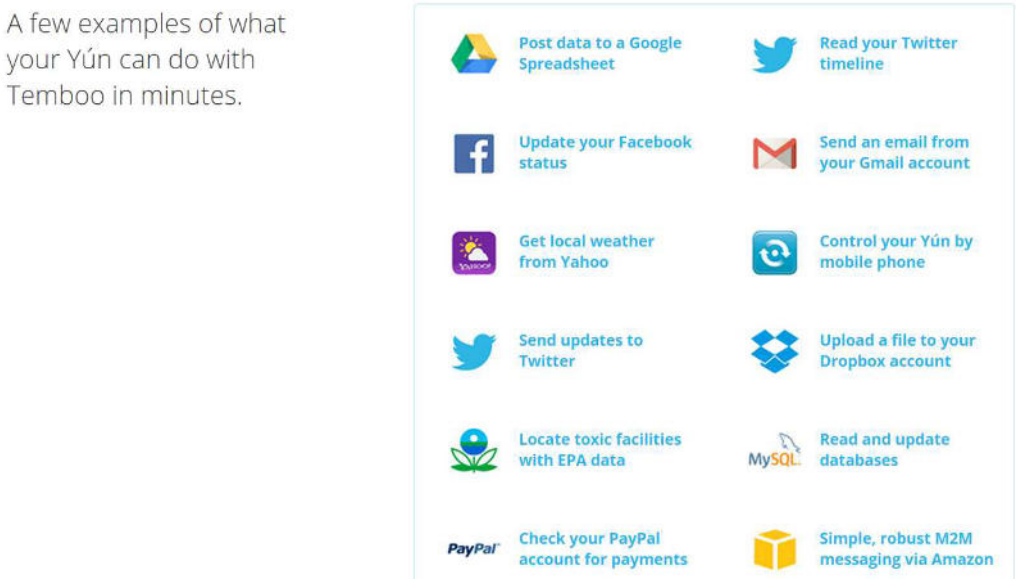


Figura 28 - Aplicações do Temboo (Temboo)

Associando o DropBox ao Temboo, com um limite grátis de 250 chamadas por mês, seria necessário enviar os dados todos no final do dia (dando um total máximo de 31 chamadas por mês). Assim seria possível diminuir do tempo de aquisição de dados, em detrimento do acesso a dados em tempo real.

O DataHero (Figura 29) é um serviço *cloud* que permite a importação de ficheiros Excel e CSV de outros serviços *cloud*, tal como o Dropbox e fornece ferramentas para análise dos dados, como a construção de gráficos. Assim, associando os 3 serviços seria possível ter um armazenamento dos dados com visualização, mas não em tempo real e de forma mais complexa que o Google Drive apenas.

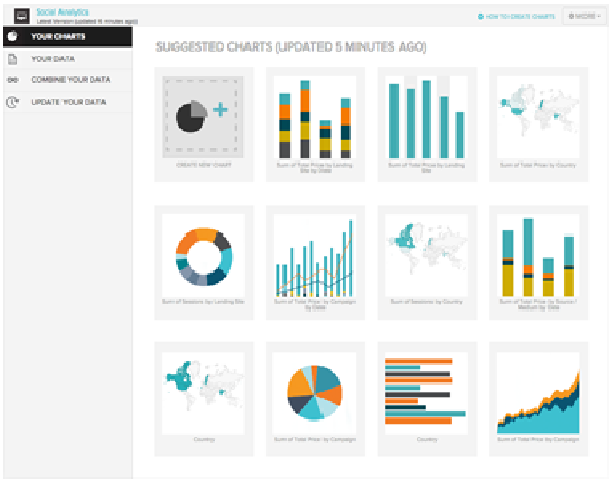


Figura 29 - Gráficos Data Hero (Dropbox)

Similar ao Data Hero, o Plotly (Figura 30) é um serviço *cloud* que fornece ferramentas para análise e visualização dados. Os dados podem vir de documentos externos como por exemplo os *datasheets* da Google Drive ou podem ser carregados diretamente no Plotly. Existe uma API que permite ser instalada no Arduino Yun e fazer a interligação destes em tempo real. O ponto negativo deste sistema é a versão gratuita só permitir neste momento a criação de um gráfico o que para este projeto é insuficiente por trabalhar com 12 sensores.

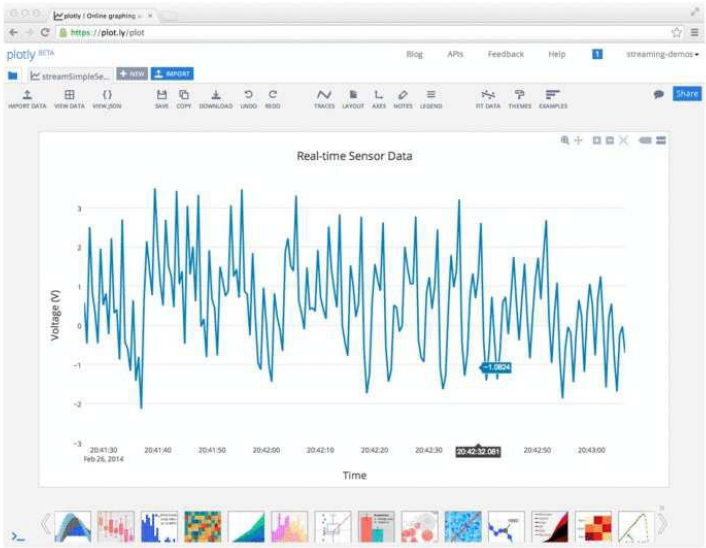


Figura 30 - Plotly (Plotly)

Face ao exposto a solução *cloud* escolhida foi a do Google Drive, por ser um serviço completamente grátis e integrado e possuir as ferramentas necessárias para os objetivos do projeto.

4.2.3.2 - Solução com servidor web do Yun

O Arduino Yun vem com um servidor web pré-instalado. Assim, uma solução para a disponibilização dos dados remotamente pode basear-se no acesso a este servidor web externamente, usando a rede do IPG através de um ip externo. Havendo a possibilidade de instalar uma base de dados no Linino para armazenar os dados e alojar um site com ferramentas para a sua análise, esta solução poderia trazer como grande vantagem a redução substancial do período entre aquisição de dados, visto, neste caso não haver necessidade de estabelecer ligação com serviços externos. A desvantagem poderia passar por sobrecarregar o Arduino Yun, mas como o acesso é limitado a um pequeno número de utilizadores essa questão não parecia ser relevante. Por outro lado a existência no Yun do cartão SD, com a informação a analisar guardada nos ficheiros.txt podia permitir outras possibilidades de manipulação dos dados.

4.3 Desenvolvimento da solução Google Cloud

Para a implementação do acesso remoto ao sistema pela solução do Google Drive, é necessário enviar os dados para uma *datasheet* no Google Drive. Posteriormente os dados poderão ser visualizados na forma de tabela ou de gráfico. Assim, foi necessário criar uma conta Google que foi identificada como simonic2pvt@gmail.com.

Para o envio dos dados teve de ser decidido se os mesmos eram enviados de uma só vez ou individualmente. O envio individual dos dados de cada sensor iria causar um maior numero de transmissão de dados o que provocaria um maior tempo de transmissão. Também teria de ser desenvolvido um *script* para colocar os dados na

posição correta, o que também aumentaria o tempo de transmissão. Por este motivo optou-se pela transmissão dos dados de uma só vez.

Seguidamente teve que se decidir como seriam enviados os dados. Como já foi referido, o Yun permite incorporar a biblioteca denominada Temboo que facilita esse processo. Ela é usada em muitos projetos analisados, mas as suas limitações na versão gratuita, não são compatíveis com o envio de muitas mensagens diárias, necessárias para uma análise em tempo real dos dados. Como o Arduino Yun possui instalado uma distribuição Linux optou-se então por usar o comando CURL. CURL é uma ferramenta para sistemas Linux e Unix que permite enviar dados para um servidor utilizando um dos protocolos suportados (HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, *LDAP ou FILE*). Neste caso a comunicação com o Google Drive terá de ser através do protocolo HTTPS. Com o comando CURL os dados serão enviados juntamente com o endereço URL. Para isso teve que ser criado um formulário no Google Drive, com os campos pretendidos para preencher o *datasheet* dos dados. A Figura 31 seguinte mostra o formulário dos dados:

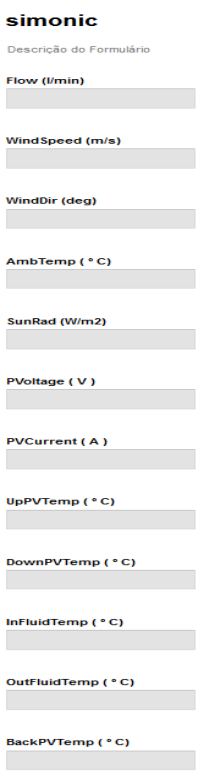


Figura 31 - Formulário responsável por carregar as *datasheets* do Google Drive

Face ao número de valores a enviar, verificou-se haver um problema com o tamanho da mensagem. O comando CURL a ser enviado tem o aspeto seguinte:

URL=https://docs.google.com/forms/d/1Vws8eA92Q80NrB42XGgNJvKivrD0Ls1IpqN0hDrguds/viewform?entry.693712574=0&entry.1794141597=0&entry.497541168=0&entry.191826194=0&entry.539550095=0&entry.1966332272=0&entry.32496276=0&entry.929025155=0&entry.1360105562=0&entry.529057279=0&entry.2006005857=0&entry.796588297=0

p.runShellCommand(curl \"URL" &submit=Submit\" -k")

O envio de um comando tão extenso traz problemas de sobrecarregamento no *buffer* que faz a comunicação entre o Arduino e o Linino. Este problema foi analisado no trabalho apresentado em (GitHub 1) onde foi apresentada a Tabela 16.

Tabela 16 - Tempo que leva a ser enviado um comando cURL utilizando Process.run() (GitHub 1)

Request URL Length (chars)	Process Speed	Correctly Executed
0 - 350	Pretty fast	Always
350 - 490	Really slow	Always
490 and higher	Super fast	Never

Como se pode verificar na Tabela 16, mensagens superiores a 490 caracteres sobrecarregam o *buffer*, fazendo com que estas nunca cheguem correctamente ao destino. De forma a minimizar problemas com o *buffer* foi utilizada a biblioteca avr/pgmspace.h que permite transmitir parte do comando através da memória ram, evitando assim sobrecarregar o *buffer* entre o Arduino e o Linino. Após este processo foi feita uma análise do tempo requerido para o sistema executar cada tarefa. Foi usada a função millis() para obter várias amostras do tempo que cada operação consumia e feita uma média obtendo a Tabela 17.

Tabela 17 - Tempo médio de processamento em cada tarefa do sistema

Operação	Receber mensagem	Tratar dados	Escrever no cartão	Enviar para Google (340 chars)	Total
Tempo em milissegundos	71	2	245	11658	11976

Como se pode concluir da Tabela 17, o tempo de envio dos dados, correspondente aos 12 sensores, para o serviço Google Drive é de longe o que consome mais tempo. Este tempo vai trazer implicações na leitura dos sensores, impossibilitando ter intervalos de leituras inferiores a 12 segundos

Finalmente os dados dos 12 sensores são apresentados na *spreadsheet*, como é mostrado na Figura 32. Na introdução dos dados o serviço da Google cria uma coluna correspondente ao tempo de receção dos dados.

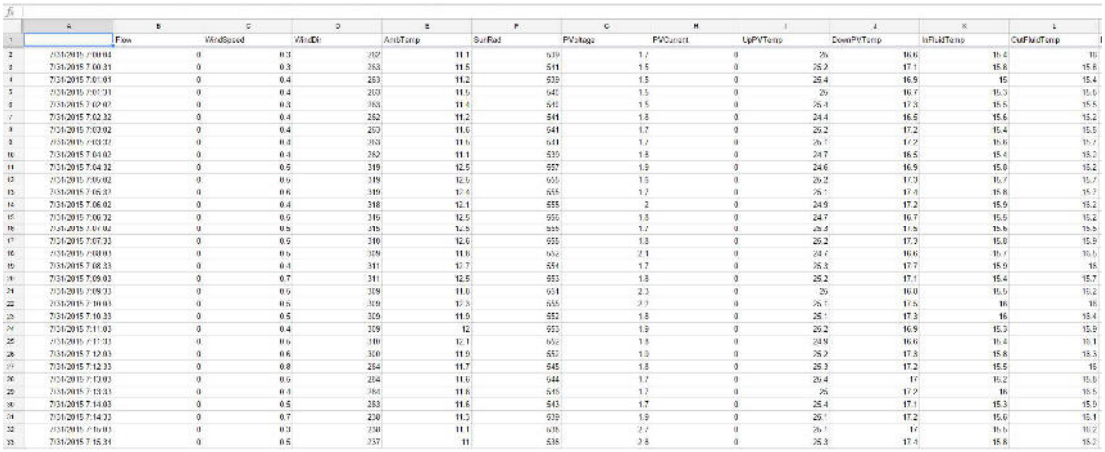


Figura 32 - *Printscreen* da tabela de dados no Google Drive

De seguida foram adicionados 12 páginas ao *datasheet* de forma a apresentar graficamente os valores obtidos por cada sensor. A Figura 33 mostra o aspeto de um gráfico (sobre o qual é possível fazer zoom).

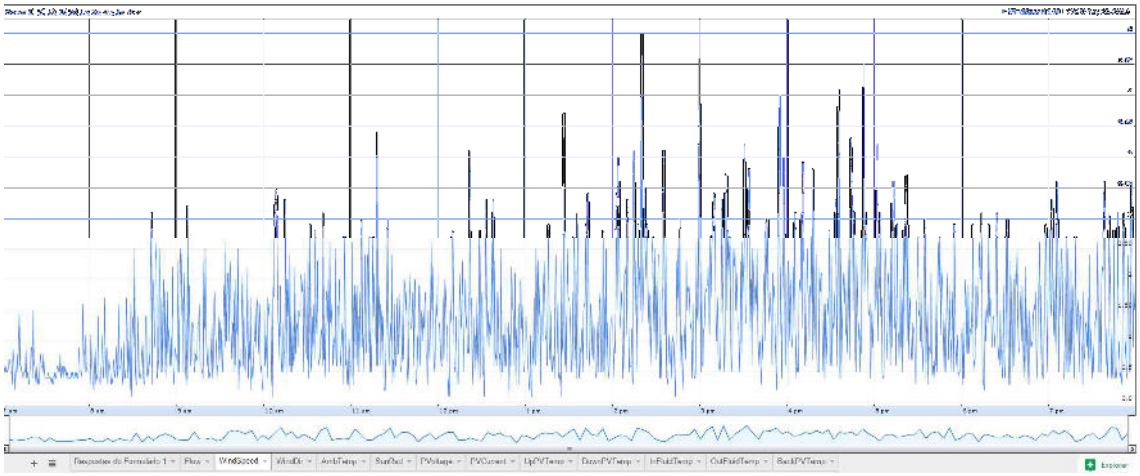


Figura 33 - *Printscreen* do gráfico de dados no Google Drive

Tendo o *datasheet* base para a receção dos dados e tendo em conta que estes se renovam todos os dias, decidiu criar-se um *datasheet* para cada dia. Para tal foi desenvolvido um *script* no Google Drive semelhante ao apresentado antes relativo ao

armazenamento de dados no cartão de memória SD. A Figura 34 apresenta o fluxograma que mostra como o Google Drive vai manipular os ficheiros de dados.

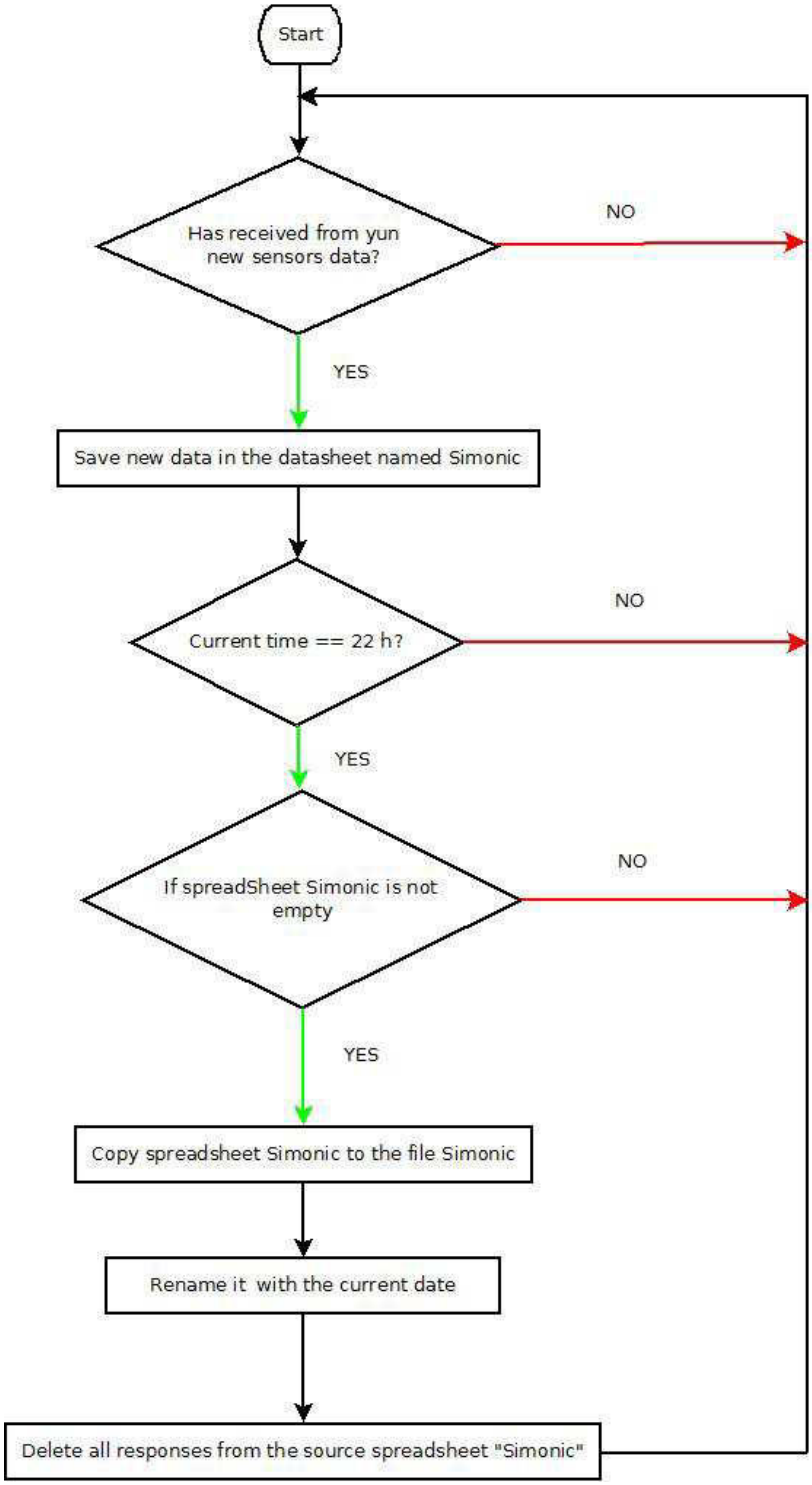


Figura 34 - Fluxograma dos *scripts* desenvolvidos no Google Drive

Como pode ser analisado no fluxograma, ao final de cada dia vai ser verificado se o ficheiro modelo contém respostas e, sendo o caso, será feita uma cópia deste ficheiro para uma pasta específica e renomeado com a data da aquisição dos dados. Por fim serão eliminadas todas as respostas do ficheiro base de modo a estar pronto para a próxima aquisição de dados. O código do script é o seguinte:

```
function copyOfDay() {
        var delay=60000;
        var name="simonic";

        var aData=Utilities.formatDate(new Date(), "GMT+0", "yyyyMd");
         name=aData+name;                    //novo nome para o ficheiro
        var ss = SpreadsheetApp.getActiveSpreadsheet();
        var s1 = ss.getSheetByName('Respostas do Formulário 1');

        var optNumRow = s1.getLastRow();    //verifica quantas dados foram introduzidos  nesse dia
         if(optNumRow>1){                   //caso existam dados
                 ss.copy(name);             //faz uma copia do spreadSheet  e dp formulário do dia
                deleteAllResponses();       //apaga todos os dados do formulario
                clearResponsesSheet();      //apaga todos os valores do spreadsheet original

                 Utilities.sleep(delay);
                copyToFolder(name);         //copia o novo spreadsheet  para uma pasta especifica
                Utilities.sleep(delay);
                deleteCopyForm();           //apaga cópia de formulário


 }
```

Tanto a pasta que contem os *datasheets* diários como o *datasheet* base com os dados em tempo real, são partilhados com as pessoas a quem for fornecido os respetivos *links*.

Com esta solução, foram alcançados os principais objetivos pretendidos para o sistema, conseguindo-se ter acesso remotamente aos dados dos sensores, em tempo (quase) real e com uma interface acessível que permite uma fácil analise dos dados. Contudo, analisando a tabela do tempo requerido por cada operação, pode concluir-se que o ponto fraco desta solução está no tempo gasto no envio dos dados para a Google. Esta limitação levou a que se a procurasse uma solução que pudesse reduzir o intervalo entre aquisição de dados, mas tal não foi possível no âmbito do Google Drive.

4.4 Desenvolvimento da solução web do Arduino Yun

O recurso ao servidor web do Arduino Yun poderia ser feito através do recurso a uma base de dados, mas foi decidido abdicar desta solução e das funcionalidades associadas em detrimento de uma menor utilização de processamento do Arduino Yun. Para isso foram aproveitados os ficheiros com dados, já existentes no cartão de memória SD.

A solução passou por desenvolver um *site* que seja carregado com os dados do dia que se pretende analisar e que as operações de filtragem ou tratamentos de dados sejam feitas do lado do cliente, libertando assim o Arduino Yun desse peso.

Como já foi referido, o Arduino Yun tem instalado um servidor web, que nos permite disponibilizar ficheiros html remotamente. Para o desenvolvimento do site irá ser preciso executar algumas operações do lado do servidor, como o acesso aos ficheiros com os dados dos sensores e para esse efeito será utilizado a linguagem PHP. O servidor web do Yun não vem com o interpretador PHP instalado e por isso o primeiro passo foi instalar as bibliotecas de PHP e fazer a sua configuração.

Para instalar o PHP foi utilizado o programa PuTTY (putty) para aceder à linha de comandos do Linux. Na linha de comandos são executados os seguintes comandos para instalar o PHP (Arduino 4, 2014).

```
opkg update
opkg install php5-cgi
```

Após a instalação do PHP deve retirar-se o comentário da seguinte linha do ficheiro php.ini:

```
list interpreter ".php=/usr/bin/php-cgi"
```

Para finalizar é preciso reiniciar o serviço UHTTPD com o seguinte comando:

```
/etc/init.d/uhttpd restart
```

Após a configuração do PHP foi desenvolvido um site para permitir ao utilizador consultar os dados existentes no cartão SD do Arduino Yun. Para o desenvolvimento do site foi utilizado um template gratuito (Figura 35) Curve desenvolvido pela ChocoTemplates (ChocoTemplates). O template foi escolhido por ser visualmente apelativo e ser otimizado para dispositivos móveis.

Figura 35 - Template Curve desenvolvido pela ChocoTemplates

O site está dividido em três partes distintas

- Página principal
- Página Data
- Página Charts

Na página principal tem-se acesso a um calendário onde é selecionado o dia que se pretende analisar. Para a criação do calendário foi utilizado o plug-in "FullCalendar" (FullCalendar). Este plug-in permite a criação de calendários e de eventos. Os eventos vão ser criados para os dias que existirem registos da leitura dos sensores. De forma a verificar em que dias houve leitura de dados o site vai fazer uma procura num local específico do cartão SD por ficheiros TXT ou CSV. Cada um desses ficheiros corresponde às leituras de um dia específico. O dia a que corresponde cada ficheiro é

feito pelo nome do ficheiro. Existem dois tipos de ficheiros, os que o nome é a data correspondente à leitura dos sensores, no formato aaaammdd, ou o ficheiro com o nome "datalog" que corresponde à leitura em tempo real caso o sistema esteja ativo nesse momento. O código corresponde à criação dos eventos do calendário é o seguinte:

```
events: [
        <?PHP
                $path = '/../';  // localização dos ficheiros com as leituras dos sensores
                $dir=opendir(__DIR__ . $path) or die("can´t open source file");
                while($file = readdir($dir)){  //le os ficheiros existentes no path
                        //ignora todos os ficheiros que não estejam no formato txt ou csv
                        if (fnmatch("*.txt",$file)||fnmatch("*.csv",$file)){
                                //cria um novo evento
                                echo "{title: 'Available',";
                                echo "value:'";
                                echo $file;
                                echo "',";
                                // caso seja seleccionado um dia altera a cor de fundo correspondente
                                        if(trataData($file)==trataData($_SESSION['varaivel_sessao'])){
                                        echo "rendering: 'background',";
                                }
                                //define a data do evento
                                echo "start:'";
                                //Chama a função trataData() para obter o dia a que o ficheiro corresponde
                                echo trataData($file);
                                echo "'},";
                        }
                }?>
]
```

O código que retira a data em que foram obtidos os dados do nome do ficheiro é

```
<?PHP
        function trataData($fname) {
                // caso o nome do ficheiro seja datalog.txt devolve a data do presente dia
                if(fnmatch("datalog.txt",$fname) ){
                        return date("Y-m-d");
                //caso contrario retira a data do nome do ficheiro
                }else{
                        $dia=substr($fname, 6,2);// retira o dia da leitura
                        $mes=substr($fname, 4,2); // retira o mês da leitura
                        $ano=substr($fname, 0,4); // retira o ano da leitura
                        $dataFinal=$ano.'-'.$mes.'-'.$dia;//constrói a data
                        return $dataFinal; // envia a data da leitura do ficheiro
                }
        }
        ?>
```

Ao seleccionar um dos dias disponíveis, o destino do ficheiro correspondente é guardado numa variável de sessão de forma a saber onde buscar os dados para a criação da tabela e do gráfico. Embora os ficheiros tenham um elevado número de registos o tamanho destes anda na ordem dos 200kB o que permite um carregamento de dados quase imediata.

O aspeto do calendário é ilustrado na Figura 36, destacando-se os dias com dados gravados e o dia selecionado para análise. Na página principal estão acessíveis acesso os botões "Data" e "Charts" que irão reencaminhar o utilizador para a tabela ou para os gráficos com os valores dos sensores carregados.

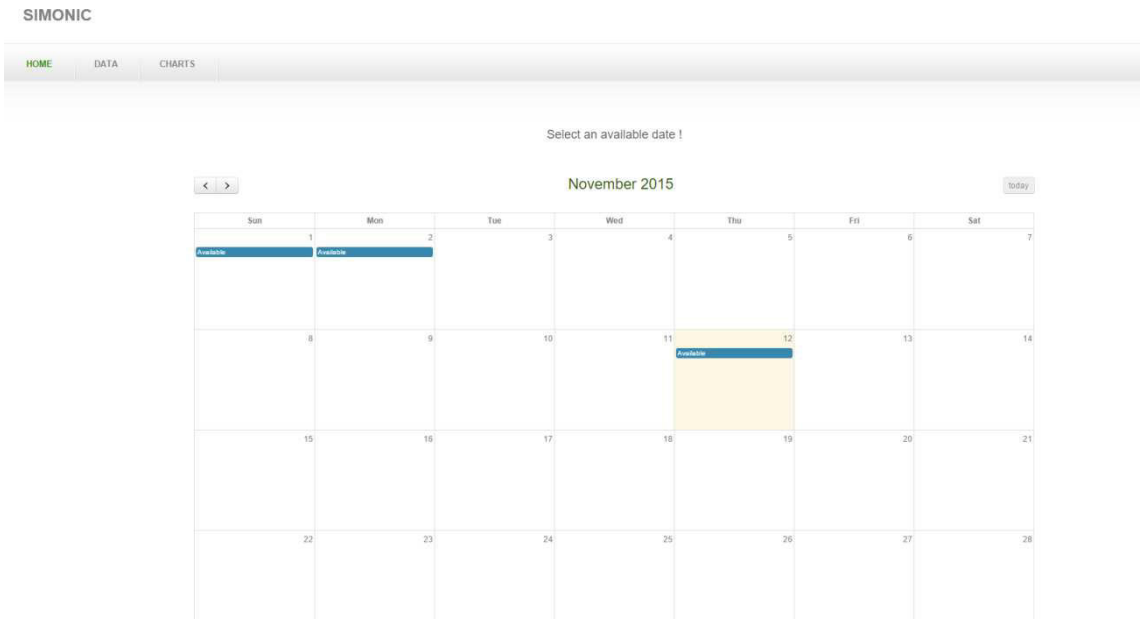


Figura 36 - *Printscreen* do calendário do site SIMONIC

Para a construção da tabela foi utilizada o plug-in "DataTables" (DataTables). Este plug-in permite criar tabelas e acrescenta várias funcionalidades úteis como, ordenação de colunas, pesquisa dentro da tabela, seleção do número de registos a serem visualizados e tem a opção de importar a tabela para vários formatos como csv, pdf e modo impressão. O aspeto da tabela é ilustrado na Figura 37.

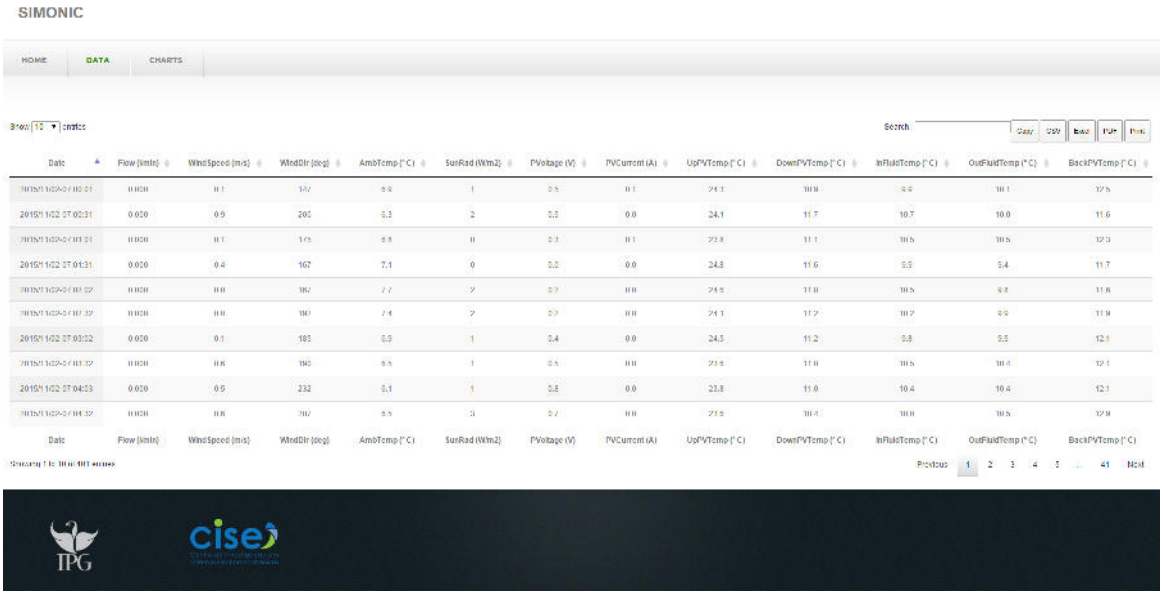


Figura 37 - *Printscreen* da tabela do site SIMONIC

Para a criação dos gráficos foi utilizada o plug-in dygraphs (dygraphs). Este plug-in permite a criação de gráficos interactivos. Existem vários plug-in's que permitem a criação de gráficos tais como n3-charts (GitHub 2), Chartist.js (GitHub 3) e ChartJS (Chart.js), mas a escolha recaiu sobre o dygraphs por esta ser gratuita, ter as ferramentas pretendidas para os gráficos que se pretendem criar, ter um visual limpo e *user friendly* e ter a peculiaridade de permitir a criação de gráficos a partir de ficheiros csv e txt. O aspeto de um gráfico é ilustrado na Figura 38:

Figura 38 - *Printscreen* do gráfico do site SIMONIC

Ao contrário do que acontecia na solução desenvolvida com recurso ao Google Drive, que só permite a consulta de um gráfico por sensor, neste caso podem ser visualizados vários sensores dentro do mesmo gráfico, o que facilita a comparação entre estes. Para isso basta seleccionar nas *checkbox's* existentes na parte superior do gráfico os sensores que se pretende visualizar. Esta funcionalidade é muito útil para comparar variáveis com ordens de grandeza similares, como a temperatura por exemplo. Da mesma forma que na solução desenvolvida com o Google Drive é possível ampliar uma área selecionada do gráfico. O acesso remoto ao site e à informação que ele contém é definido através de um endereço externo à rede do IPG.

Em comparação com a solução Google, os tempos totais de processamento são muito reduzidos conforme se confirma com os dados constantes da Tabela 18:

Tabela 18 - Tempo médio de cada operação utilizando o servidor web do Arduíno Yun

Operação	**Receber mensagem**	**Tratar dados**	**Escrever no cartão**	**Total**
Tempo em milissegundos	71	2	245	318

4.5 Controlo remoto do sistema

Outro dos requisitos para este projeto é a possibilidade de controlar o sistema remotamente. O controlo pretendido é ode ligar e desligar o sistema e de reconfigurar o período de aquisição de dados. Para esse efeito foi desenvolvido um site para a utilização unicamente do administrador do sistema. Este, podia ter sido integrado no site desenvolvido para a consulta de dados, mas visto existirem duas soluções distintas, não faria grande lógica na utilização da solução do Google Drive ter de se aceder à segunda solução para controlar o sistema.

Para o desenvolvimento do site de controlo remoto foi utilizado novamente o Template Curve utilizado no site anterior. Para aceder ao site do administrador é requerido uma *password*, e para aumentar a sua segurança foi encriptada utilizando o algoritmo md5 (PHP.net). A janela de entrada deste site é ilustra na Figura 39:

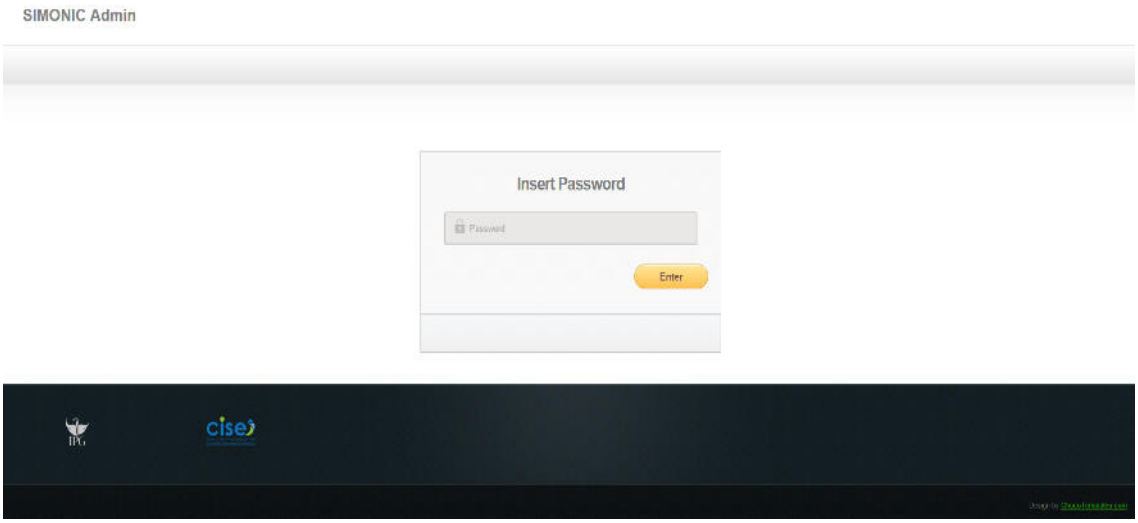


Figura 39 - *Printscreen* da página de acesso ao site SIMONIC Admin

O site de administrador tem como funcionalidades

- Ligar/desligar o sistema
- Alterar o período de aquisição de dados
- Alterar a *password* de acesso ao site

O código seguinte mostra as configurações necessárias para esse efeito.

```
// Yun server
YunServer server;
void setup() {
        server.listenOnLocalhost();
        server.begin();
}
void loop() {
        // Get clients coming from server
         YunClient client = server.accept();
         // There is a new client?
         if (client) {
           // Process request
           process(client);
           // Close connection and free resources.
           client.stop();
         }
        …
}
```

Para este controlo é necessário que o Arduino Yun fique à escuta de comandos remotos, o que é feito com o código seguinte:

```
void process(YunClient client) {
        // read the command
        char command = client.read() ;

        // is "turnOnOff" command?
        if (command == 'c') {
                // call the same function of the button installed in arduino yun to start/paused the
                //system
                turnOnOff();
                client.println(204);.
        }

        // is "readState" command?
        if (command == 'r') {
                client.println(estado);//send state of the system
        }
        case 'n': {
          volatile const int unsigned myRange[] = {5000, 15000, 30000,60000};
            int nrange = client.parseInt();
            if(nrange<4)
              readRange=myRange[nrange];
        }
        break;
        case 's': {
          client.println(readRange);
        }
        break;
        }
}
```

O envio dos comandos para o Arduino Yun é feito por url constituído por:

Endereço_IP_do_Arduino/Nome_Do_ARDUINO/Comando/variável

O comando pode ou não enviar uma variável. Um exemplo de comando para ler o estado do sistema será:

http://192.168.XXX.XXX/arduino/r/

A página do site que permite ativar/desligar o sistema, envia um comando semelhante ao anterior para saber em que estado está o sistema. A resposta a esse comando vai atualizar o estado do botão existente no site, ilustrado na Figura 40.

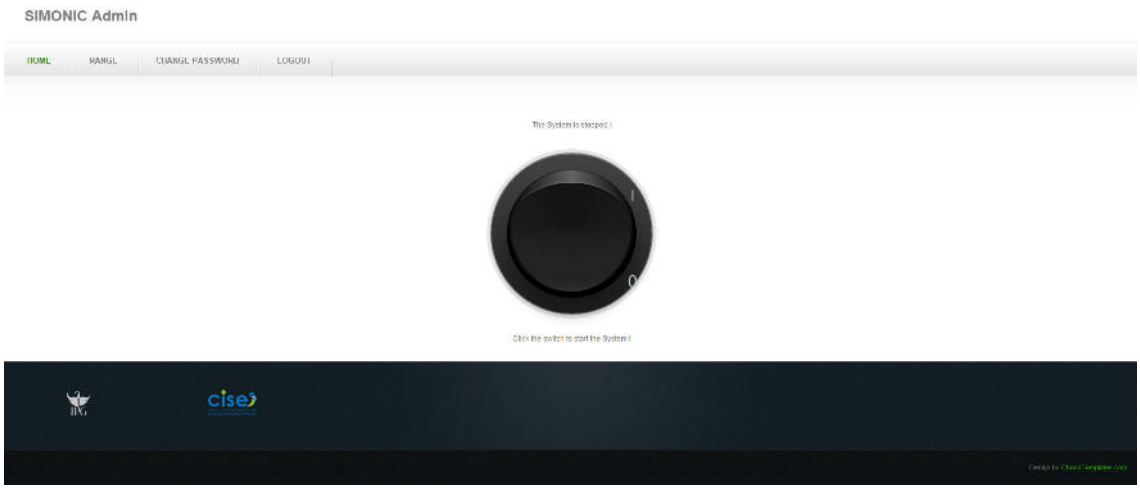


Figura 40 - *Printscreen* da página do site que permite ativar/desligar o sistema

Ao ser premido o botão este envia o comando de alterar o estado do sistema e actualiza a página do site, visível na Figura 41:

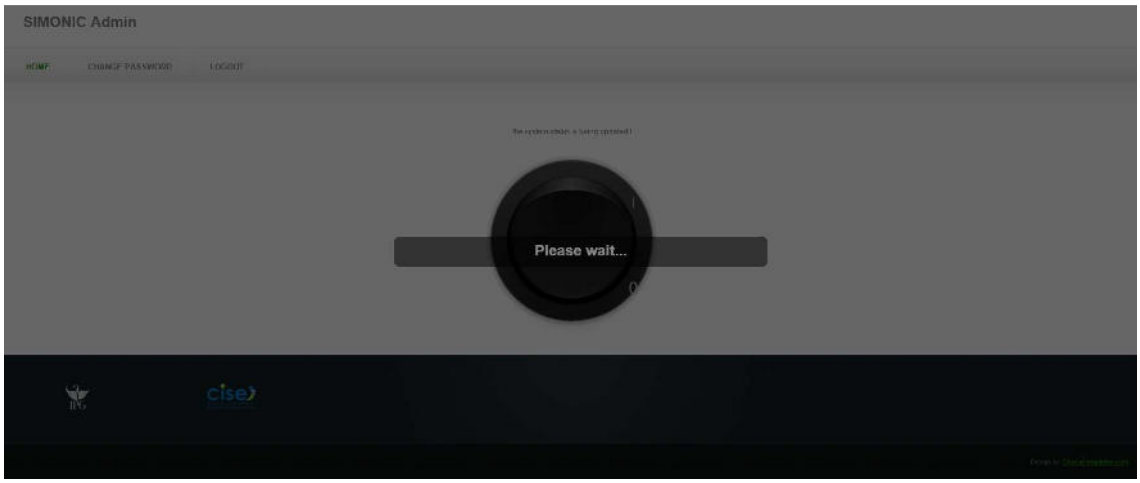


Figura 41 - *Printscreen* da página do site a atualizar estado do sistema

A página que permite alterar o período de aquisição de dados tem como particularidade poder ativar/desativar o processo de envio de dados para a Google Cloud. Isto deve-se à existência dos dois sistemas distintos para a consulta de dados, e

do sistema da Google requerer períodos de aquisições superiores. Desta forma sempre que forem definidos tempos inferiores a 30 segundos a consulta de dados só será possível através do website do Yun. A Figura 42 ilustra as opções de aquisição que são 5, 15, 30 ou 60 segundos, sendo que as 2 primeiras são executadas apenas no Yun.

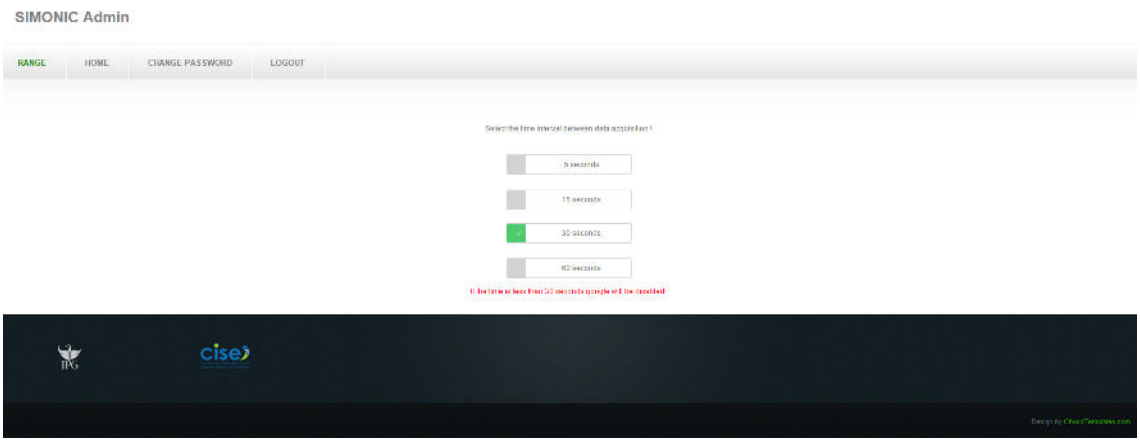


Figura 42 - *Printscreen* da página que permite configurar os períodos de aquisição de dados

Por fim existe a opção de alterar a *password* de acesso ao site de administrador (Figura 43).

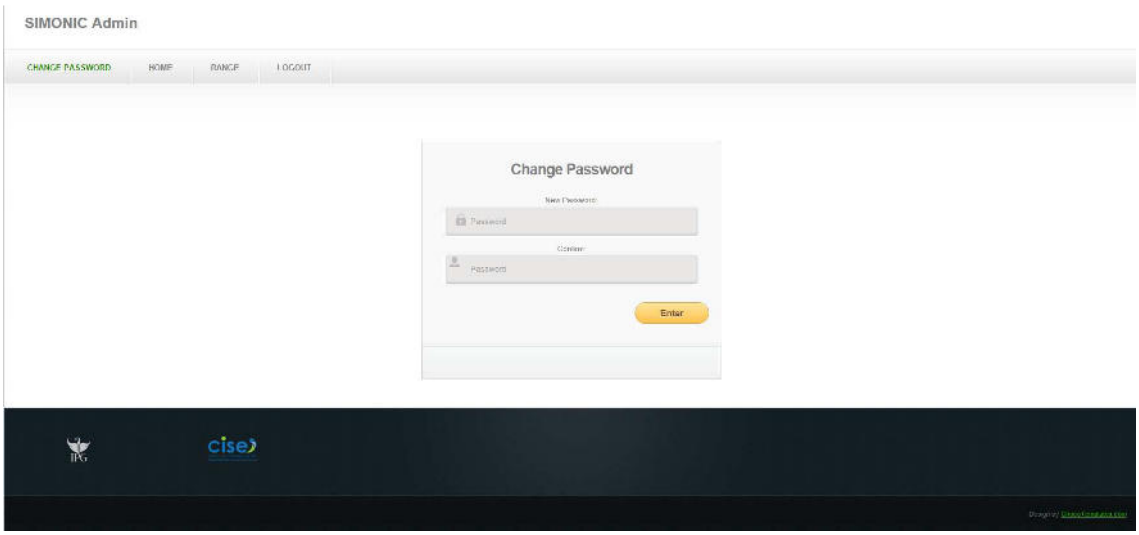


Figura 43 - *Printscreen* da página que permite alterar a *password* do administrador

4.6 Sistema de alerta

No desenvolvimento do projeto sentiu-se a necessidade da criação de um sistema de alerta para prevenir a danificação do sistema PVT. Pretende-se com este sistema alertar o utilizador responsável, através de um email, no caso da temperatura do painel atingir valores críticos que ponham em risco sistema PVT. Desta forma é monitorizada a temperatura *UpPVTemp* e caso seja atingida a temperatura máxima pré-definida será enviado um email de alerta para o utilizador responsável pelo sistema. Só será enviado um alerta por dia, salvo o sistema seja reiniciado, nesse caso poderá ser reenviado um novo alerta.

O sistema de alerta foi implementado no Arduino Yun, utilizando o serviço **Temboo** para envio do email. Este serviço já tinha sido explorado para o envio dos dados dos sensores para a *cloud*, tendo sido excluído devido à versão gratuita disponibilizar apenas 250 chamadas por mês. No entanto para o sistema de alerta as 250 mensagens são suficientes, visto só serem enviadas alertas em situações excecionais.

Para a implementação do sistema foram necessários os seguintes passos:

1. Criar conta Temboo (Temboo)
2. Ativar a autenticação (em 2 passos) através da conta gmail
3. Criar uma aplicação no Temboo (Figura 44)
4. Configurar o YUN

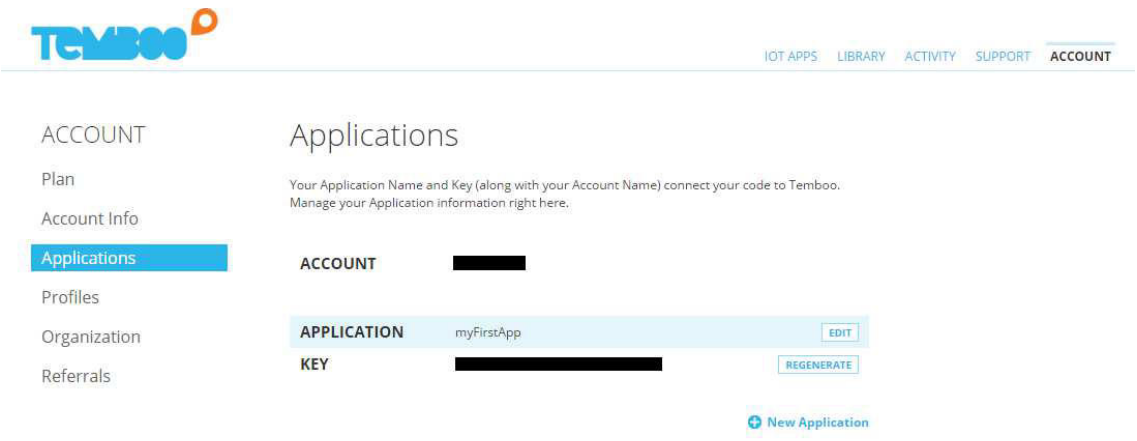


Figura 44 - Criação de uma aplicação Temboo (Temboo)

O código correspondente à criação do alerta é o seguinte:

```
void sendWarning(){

 #define TEMBOO_ACCOUNT F("xxxxxxxxxx")
 #define TEMBOO_APP_KEY_NAME F("xxxxxxxxxxx")
 #define TEMBOO_APP_KEY  F("xxxxxxxxxxxxxxxxxxxxxxx")


 #define GMAIL_USER_NAME F("xxxxxxxxx@gmail.com")
 // your Gmail App-Specific Password
 #define GMAIL_PASSWORD F("gmailPassword")

 TembooChoreo SendEmailChoreo;

  // invoke the Temboo client
  // NOTE that the client must be reinvoked, and repopulated with
  // appropriate arguments, each time its run() method is called.
  SendEmailChoreo.begin();

 // set Temboo account credentials
  SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT);
  SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
  SendEmailChoreo.setAppKey(TEMBOO_APP_KEY);

 // identify the Temboo Library choreo to run (Google > Gmail > SendEmail)
  SendEmailChoreo.setChoreo(F("/Library/Google/Gmail/SendEmail"));


  // the first input is your Gmail email address
  SendEmailChoreo.addInput(F("Username"), GMAIL_USER_NAME);
  // next is your Gmail App-Specific password.
  SendEmailChoreo.addInput(F("Password"), GMAIL_PASSWORD);
  // who to send the email to
  SendEmailChoreo.addInput(F("ToAddress"), GMAIL_USER_NAME);
  // then a subject line
  SendEmailChoreo.addInput(F("Subject"), F("ALERT!"));

   // next comes the message body, the main content of the email
  SendEmailChoreo.addInput(F("MessageBody"), F("Temp. too high!"));

  // tell the Choreo to run and wait for the results. The
  // return code (returnCode) will tell us whether the Temboo client
  // was able to send our request to the Temboo servers
  SendEmailChoreo.run();
 SendEmailChoreo.close();
 delay(15000);
}
```

A mensagem de alerta recebida tem o aspeto ilustrado na Figura 45.

Figura 45 - Email de alerta

Atendendo a que o alerta baseado no email necessita de uma ligação à internet para a sua receção, foi também idealizado um sistema de alerta para envio de um SMS através da rede GSM, para permitir a receção do alerta em qualquer lugar usando um vulgar telemóvel. Inicialmente estava previsto usar um serviço grátis do gmail para o efeito. Contudo esse serviço foi descontinuado em 1 de Abril 2015, pelo que se procuraram alternativas. Existem várias disponíveis para o efeito, mas com muitas limitações a nível da quantidade total de SMS enviados de forma grátis. O próprio serviço Temboo (Temboo) usado para o envio do email pode ser associado com outro serviço, o Twillio (Twilio), para envio de SMS. Porém esse serviço em Portugal é pago. Foram pesquisados outros serviços smspubli (smspubli), FreeSMS (freesms), IFTTT (ifttt), SMSGlobal (smsglobal), Messagebird (messagebird), entre outras, com opções de teste grátis, mas nalguns casos a quantidade de SMS grátis era muito reduzida e rapidamente atingida nos testes e noutros casos não funcionava com números portugueses. Devido aos motivos a cima referidos o sistema de alerta por SMS não foi implementado.

Concluído o desenvolvimento das soluções concebidas para a obtenção dos objetivos propostos no início do projeto, o sistema foi testado de forma mais extensiva, mantendo-o a funcionar de forma consecutiva durante várias semanas.

Capitulo 5 - Testes e resultados

O teste do sistema teve por base a verificação da sua integridade assim como a validade dos dados adquiridos.

Para testar a validação dos dados foram comparadas amostras dos valores dos sensores de três formas diferentes:

- Diretamente nalgumas variáveis com recurso sensores específicos externos;
- Utilizando o sistema inicial em que os dados são recebidos por cabo no computador do laboratório e analisados pela aplicação MatLab;
- Utilizando o sistema desenvolvido

A validação dos resultados tem em conta sobretudo a comparação entre os valores obtidos com a aplicação cablada e o MatLab, já usada para outros estudos e os valores obtidos com a nova solução sem fios e os Arduinos.

Como exemplo, as figuras seguintes mostram consecutivamente o gráfico obtido usando os dados recolhidos em 25/06/2015 para a radiação solar e corrente elétrica, através do MatLab e tratamento gráfico no Excel Figura 46, através do Google Drive (Figura 47) e do servidor web com o dygraph (Figura 48).

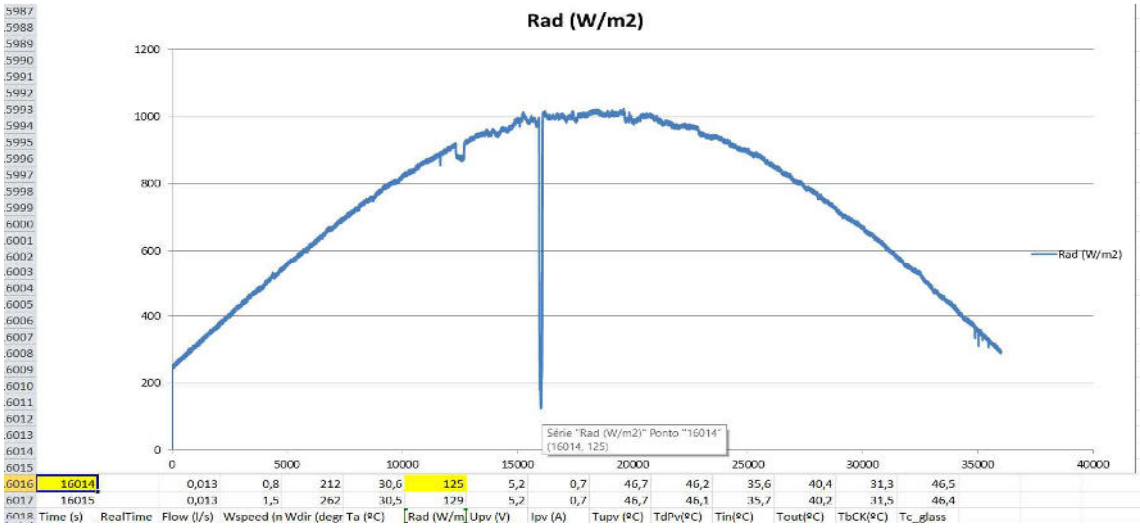


Figura 46 - Gráfico criado com os dados adquiridos pelo MatLab (Vieira, Ramos, Cardoso, Saraiva, & Alcaso, 2015)

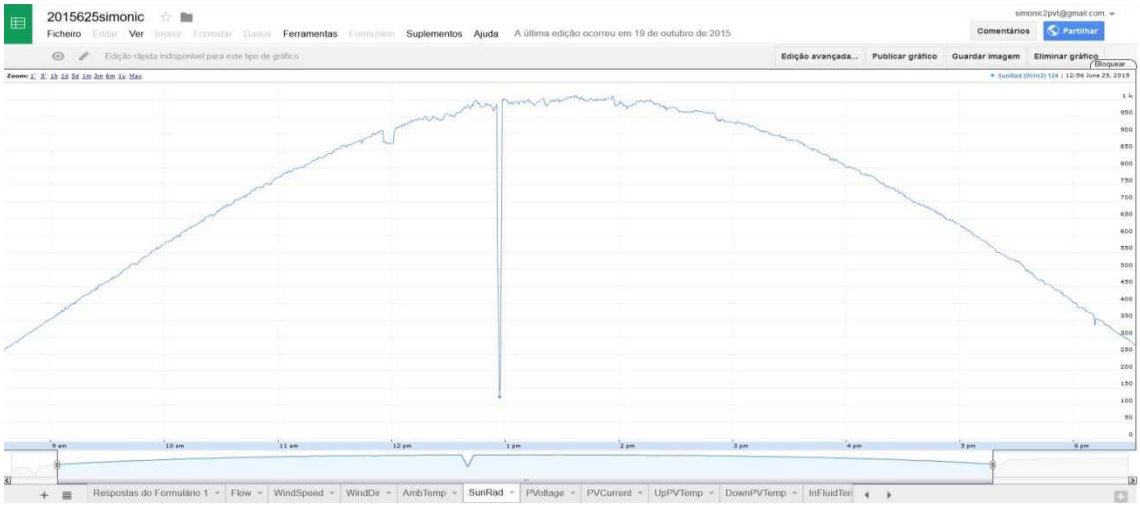


Figura 47 - Gráfico criado pelo Google Docs

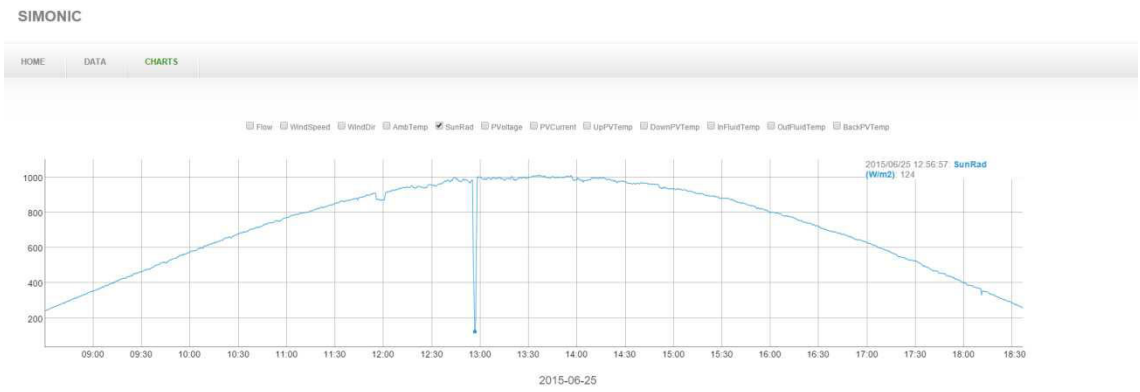


Figura 48 - Gráfico criado com dygraph

Para uma melhor comparação dos resultados a Figura 48 mostra os resultados do MatLab e Google no mesmo gráfico traçado em Excel. Apesar da diferença de tempos de aquisição (1 e 30 segundos respetivamente) não parecem existir diferenças substanciais. Contudo, uma ampliação dos gráficos ilustrada na Figura 49 revela que a curva do MatLab possui mais "ruído" como seria de esperar. Verifica-se ainda que ambas as curvas têm uma evolução similar, havendo uma pequena diferença nos valores o que se explica pelos conversores ADC do Arduino e computador serem diferentes.

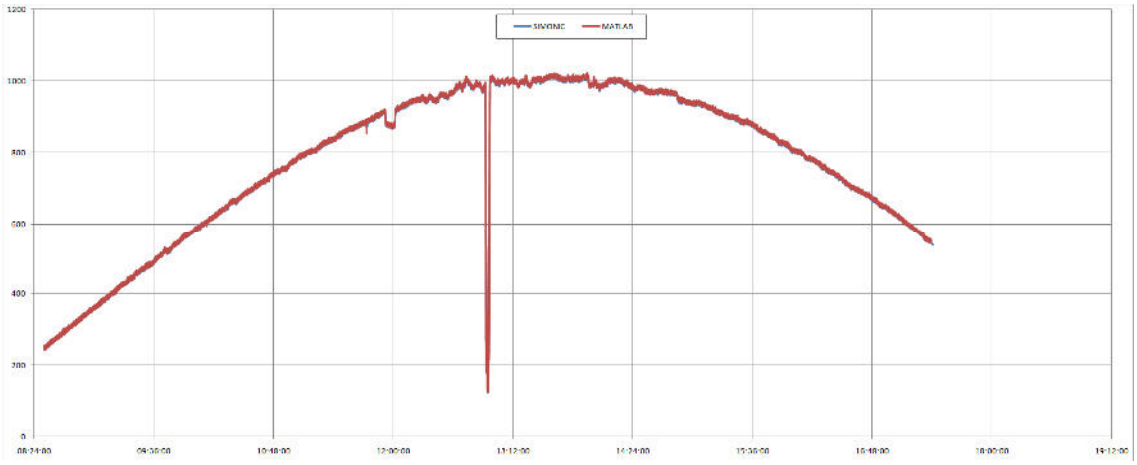


Figura 49 - Gráfico comparativo MatLab-Google

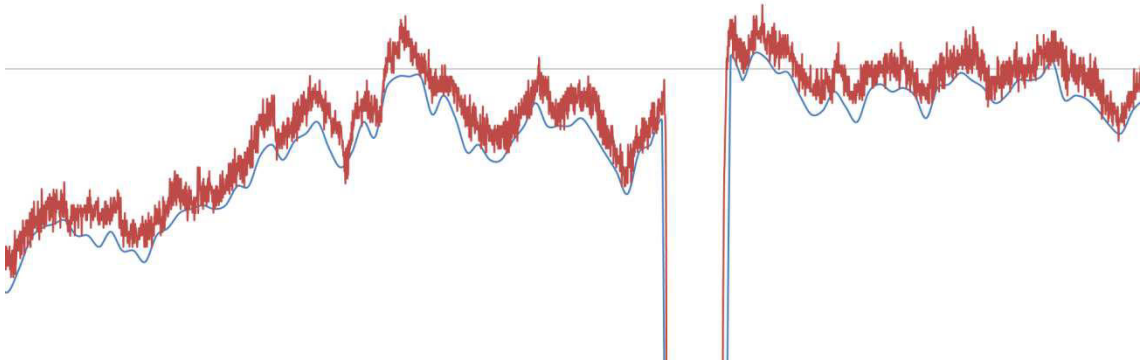


Figura 50 - Comparativo MatLab-Google ampliado

Para testar a integridade do sistema este foi deixado activo durante vários dias consecutivos. O sistema comportou-se como era esperado, com a excepção de quando houve falhas de ligação à internet exterior ou falhas de energia elétrica. Nesses casos o sistema deixava de enviar dados para o serviço da Google. No caso de falha na ligação à internet, mesmo após a reposição do serviço os dados não eram enviados, o que se devia ao facto de o Arduino Yun criar a sua própria rede internet, quando não detetava outra rede. A solução passava por fazer o reset do Arduino. Porém, os dados existiam no cartão SD, pelo que não era perdida informação. No caso da falha de energia apenas, o Arduino costumava retomar o envio dos dados não havendo contudo qualquer informação relativa ao período de falha de energia, como seria de esperar.

Capitulo 6 – Conclusão

Neste relatório apresentou-se a forma como foi desenvolvida a solução para monitorização remota do sistema PVT tendo em conta as suas características e ambiente envolvente e procurando minimizar custos de desenvolvimento e exploração. Foram desenvolvidas duas soluções distintas redundantes/complementares, uma baseada no Google Drive e outra usando o servidor web do Arduino Yun. Esta última opção surgiu, ao longo do desenvolvimento do trabalho, como solução para ultrapassar uma limitação do Google Drive associada ao intervalo de aquisição de dados. Embora não seja usado um serviço *cloud* tradicional, o servidor Web consegue dar acesso remoto aos dados, tendo em consideração que esta solução é possível desde que a quantidade de utilizadores a aceder ao servidor seja relativamente reduzida, visto o Arduino Yun ser, neste aspeto, um sistema com recursos relativamente reduzidos. No futuro, o uso de novas plataformas microcontroladoras com velocidades de processamento superiores, atenuará aquela limitação. Também o uso de ligações Wi-Fi e servidores *cloud* mais rápidos permitirá tornar o processamento na rede mais próximo do tempo real, que é uma condição desejada em sistemas de aquisição de dados. Note-se que as limitações do Yun foram também visíveis na sua gestão de memória, que obrigava à restruturação e otimização do código cada vez que havia necessidade de introduzir novas funções ao sistema.

No âmbito da solução desenvolvida poder-se-ia pensar futuramente em incorporar no servidor web do Yun (ou de preferência numa plataforma similar mas com maior velocidade e capacidade), uma base de dados do tipo SQL. Esta solução permitirá lidar mais facilmente com a grande quantidade de dados que se acumulará com o tempo, extraindo a informação de forma mais seletiva. Aliás esta questão é muito importante para o desenvolvimento de soluções similares para outros sistemas de energias renováveis já instalados e a instalar no parque de energias renováveis da ESTG.

A capacidade do envio de SMS seria também uma opção interessante, desde que sem custos.

De salientar por fim a importância deste estágio como etapa de aprendizagem de temas não relacionados diretamente com as tecnologias de informação e informáticas e da verificação da importância que estas têm nos mais variados campos quando associadas a sistemas embebidos.

Referências

123RF. (s.d.). Obtido em Dezembro de 2015, de 123RF: http://es.123rf.com/profile_cheskyw

aiceo Portugal Global. (Setembro de 2015). *Portugal - Ficha País.* Obtido em Dezembro de 2015, de http://www.portugalglobal.pt: http://www.portugalglobal.pt/pt/biblioteca/livrariadigital/portugalfichapais.pdf

Alecrim, E. (23 de Dezembro de 2008). *O que é cloud computing (computação nas nuvens)?* Obtido em Dezembro de 2015, de infowester: http://www.infowester.com/cloudcomputing.php

Arduino 1. (s.d.). *Arduino Products.* Obtido em Dezembro de 2015, de Arduino: https://www.arduino.cc/en/Main/Products

Arduino 2. (s.d.). *Arduino Yún.* Obtido em Dezembro de 2015, de Arduino: https://www.arduino.cc/en/Main/ArduinoBoardYun

Arduino 3. (s.d.). *attachInterrupt()*. Obtido em Junho de 2015, de Arduino: https://www.arduino.cc/en/Reference/AttachInterrupt

Arduino 4. (2014). *Topic: PHP / uhttpd*. Obtido em Outubro de 2015, de forum.arduino: http://forum.arduino.cc/index.php?topic=221261.msg1607985#msg1607985

Basconcello, D. O. (s.d.). *O que é o Arduino*. Obtido em Dezembro de 2015, de robotizando: http://www.robotizando.com.br/curso_arduino_o_que_e_arduino_pg1.php

Chart.js. (s.d.). *Chart.js*. Obtido em Outubro de 2015, de Chart.js: http://www.chartjs.org/

Chauvin-Arnoux. (s.d.). *Testing instrument for photovoltaic installations.* Obtido em 2 de 2016, de chauvin-arnoux: http://www.chauvin-arnoux.com/sites/default/files/D00KZN16_1.PDF

ChocoTemplates. (s.d.). *Curve - Free Template*. Obtido em Outubro de 2015, de ChocoTemplates: http://chocotemplates.com/corporate/curve/

Churchill, S. (Abril de 2007). *MaxStream ZigBee module*. Obtido em Dezembro de 2015, de dailywireless.org: http://www.dailywireless.org/2007/04/03/maxstream-zigbee-module/

CISE. (2016). *Centro de Investigação em Sistemas Electromecatrónicos*. Obtido de CISE: http://www.cise.ubi.pt/

Clariá, F. S. (2 de Novembro de 2012). *How I Made a Fully-Functional Arduino Weather Station*. Obtido em Junho de 2015, de developers: http://www.toptal.com/c/how-i-made-a-fully-functional-arduino-weather-station-for-300

DataHero. (s.d.). *DataHero*. Obtido de DataHero: https://datahero.com/

DataTables. (s.d.). *DataTables*. Obtido em Outubro de 2015, de DataTables: https://www.datatables.net/

DIGI 1. (s.d.). *Connectivity for critical infrastructures*. Obtido de digi: http://www.digi.com/

Digi 2. (s.d.). *XBee™ Gateway User's Guide*. Obtido de Digi: http://ftp1.digi.com/support/documentation/html/90001399/90001399_A/Files/XBee-concepts.html

Dropbox. (s.d.). *Dropbox*. Obtido de Dropbox: https://www.dropbox.com/

dygraphs. (s.d.). *dygraphs*. Obtido em Outubro de 2015, de dygraphs: http://dygraphs.com/

Eicker, U. (2003). *Solar Technologies for Buildings.* Wiley.

Enerdata 1. (2015). *Total energy consumption*. Obtido em Dezembro de 2015, de enerdata: https://yearbook.enerdata.net/

Enerdata 2. (2015). *CO2 emissions from fuel combustion*. Obtido em Dezembro de 2015, de enerdata: https://yearbook.enerdata.net/#CO2-emissions-data-from-fuel-combustion.html

Enerdata 3. (2015). *Share of renewables in electricity production (incl hydro)*. Obtido em Dezembro de 2015, de ENERDATA: https://yearbook.enerdata.net/#renewable-in-electricity-production-share-by-region.html

Enerdata 4. (2015). *Share of wind and solar in electricity production*. Obtido em Dezembro de 2015, de Enerdata: https://yearbook.enerdata.net/#wind-solar-share-electricity-production.html

freesms. (s.d.). *freesms*. Obtido em Janeiro de 2016, de freesms: http://www.freesms.net/

FullCalendar. (s.d.). *FullCalendar*. Obtido em Outubro de 2015, de FullCalendar: http://fullcalendar.io/

GitHub 1. (s.d.). *yun-adc-storage*. Obtido em Junho de 2015, de GitHub: https://github.com/derekcormier/yun-adc-storage/blob/master/README.md

GitHub 2. (s.d.). *n3-charts*. Obtido em Outubro de 2015, de github: http://n3-charts.github.io/line-chart/#/home

GitHub 3. (s.d.). *Chartist.js*. Obtido em Outubro de 2015, de gionkunz: https://gionkunz.github.io/chartist-js/

Google Drive. (s.d.). *Google Drive*. Obtido de Google Drive: https://www.google.com/intl/pt-PT/drive/

Iceubi2015. (2015). *iceubi2015*. Obtido de iceubi2015: http://iceubi2015.ubi.pt/?lang=pt

iCloud. (s.d.). *iCloud*. Obtido de iCloud: https://www.icloud.com/

ifttt. (s.d.). *ifttt*. Obtido em Janeiro de 2016, de ifttt: https://ifttt.com/recipes/

Kipp & Zonen. (s.d.). *CMP3 Pyranometer*. Obtido em Abril de 2015, de Kipp & Zonen: http://www.kippzonen.com/Product/11/CMP3-Pyranometer#.VrxITNCg0qt

Kumar, L. (s.d.). *Meaning of G, E, 2G, 3G, H, 4G in Mobile Internet Signal Bar*. Obtido em Dezembro de 2015, de techwelkin: http://techwelkin.com/meaning-mobile-symbols-g-e-2g-3g-h-4g-mobile-internet-signal-bar

McCoy, C. (29 de Abril de 2011). Remote Weather Station. *CEN 4935* , 32. Florida Gulf Coast University.

messagebird. (s.d.). *messagebird*. Obtido em Janeiro de 2016, de messagebird: https://www.messagebird.com/

Monk, S. (2013). *Programming Arduino Next Steps: Going Further with Sketches.* McGraw-Hill Education Tab.

Nadav Savio, J. B. (2007). Design Sketch: The Context of Mobile Interaction. *MobileHCI07 9th International Conference is Human Computer Interaction with Mobile Devices and Services* , (p. 3). Singapore.

National Instruments Corporation. (s.d.). *NI DAQCard -6062E Family Specifications.* Obtido de ni.com: http://www.ni.com/pdf/manuals/370724c.pdf

Oliveira, S. d. (2012). *CONTROLE DE DISPOSITIVOS UTILIZANDO* . Obtido em Dezembro de 2015, de professorpetry: http://www.professorpetry.com.br/Ensino/Defesas_Pos_Graduacao/Defesa%2034_Sergio%20de%20Oliveira%20_Controle%20de%20Dispositivos%20Utilizando%20Modulos%20Wifi%20Xbee.pdf

OneDrive. (s.d.). *OneDrive*. Obtido de OneDrive: https://onedrive.live.com/about/pt-pt/

Pace, N. (25 de Março de 2015). *For Earth Hour, Use Less of Everything*. Obtido em Dezembro de 2015, de huffingtonpost: http://www.huffingtonpost.com/natalie-pace/for-earth-hour-use-less-of-everything-_b_6934526.html

PHP.net. (s.d.). *md5*. Obtido em Novembro de 2015, de pho.net: http://php.net/manual/en/function.md5.php

Pires, C. (25 de Maio de 2012). *Portugal Um país com imensa energia*. Obtido em Dezembro de 2015, de Diário de Notícias: http://www.dn.pt/revistas/nm/interior/portugal-um-pais-com-imensa-energia-2546081.html

Plotly. (s.d.). *Plotly*. Obtido de Plotly: https://plot.ly/

putty. (s.d.). *Download PuTTY - a free SSH and telnet client for Windows*. Obtido em Outubro de 2015, de putty.org: www.putty.org

S. Zahurula, N. M. (2014). Development of a prototype for remote current measurements of PV panel using WSN. (p. 6). International Journal of Smart Grid and Clean Energy, vol 3, no 2, april 2014.

Santos, F. D. (9 de Junho de 2015). *Paris! Vamos desinvestir nos combustíveis fósseis?* Obtido de Publico: https://www.publico.pt/ecosfera/noticia/paris-vamos-desinvestir-nos-combustiveis-fosseis-1698371

Schwartz, M. -O. (22 de Julho de 2015). Cloud-Connected Weather Station with the Arduino Yun. *adafruit learning System*.

SMA Solar Technology. (s.d.). *SUNNY WEBBOX with Bluetooth Wireless Technology*. Obtido em Dezembro de 2015, de sma benelux: http://www.sma-benelux.com/nl_BE/producten/monitoring-systemen/sunny-webbox-met-bluetoothr/webspecial/the-new-sunny-webbox.html

smsglobal. (s.d.). *smsglobal*. Obtido em Janeiro de 2016, de smsglobal: http://www.smsglobal.com/

smspubli. (s.d.). *smspubli*. Obtido em Janeiro de 2016, de smspubli: http://smspubli.com/

Temboo. (s.d.). *Temboo*. Obtido em 2015, de Temboo: https://temboo.com/

Texas Instruments. (s.d.). *AN-162 LM2907 Tachometer/Speed Switch Building Block Applications.* Obtido em Abril de 2015, de Texas Instruments: http://www.ti.com.cn/cn/lit/an/snaa088/snaa088.pdf

Thies Clima. (s.d.). *Combined Wind Transmitte.* Obtido em Abril de 2015, de biral: http://www.biral.com/wp-content/uploads/2014/09/smallCW.pdf

Twilio. (s.d.). *Twilio*. Obtido em Janeiro de 2016, de Twilio: https://www.twilio.com/

Vieira, P. A., Ramos, C. A., Cardoso, A. J., Saraiva, L. M., & Alcaso, A. N. (2015). Development of a cloud-based system for remote monitoring of a PVT panel. *ICEUBI*, (p. 7). Covilhã.

webopedia. (14 de Julho de 2010). *Wi-Fi Definition is Not Wireless Fidelity*. Obtido de webopedia: http://www.webopedia.com/DidYouKnow/Computer_Science/wifi_explained.asp

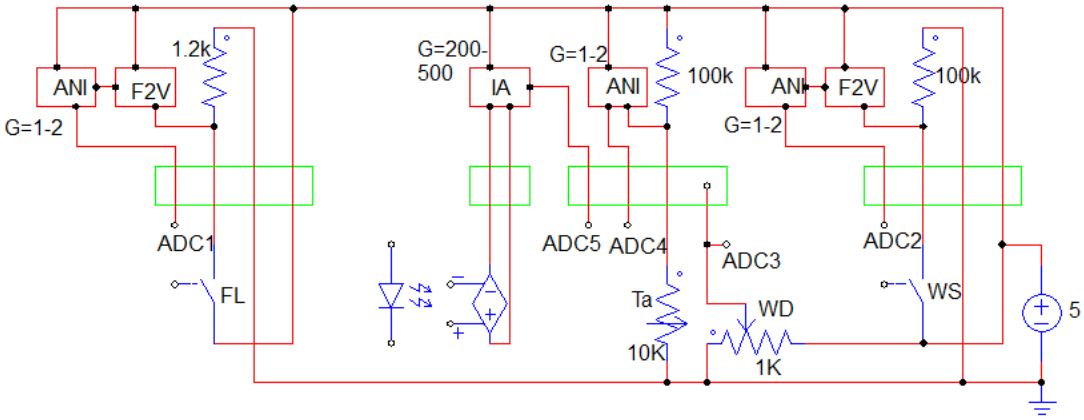
WK3. (s.d.). *Como os sites devem se adaptar aos dispositivos móveis*. Obtido em Dezembro de 2015, de wk3: http://www.wk3.com.br/blog/como-os-sites-devem-se-adaptar-aos-dispositivos-moveis/

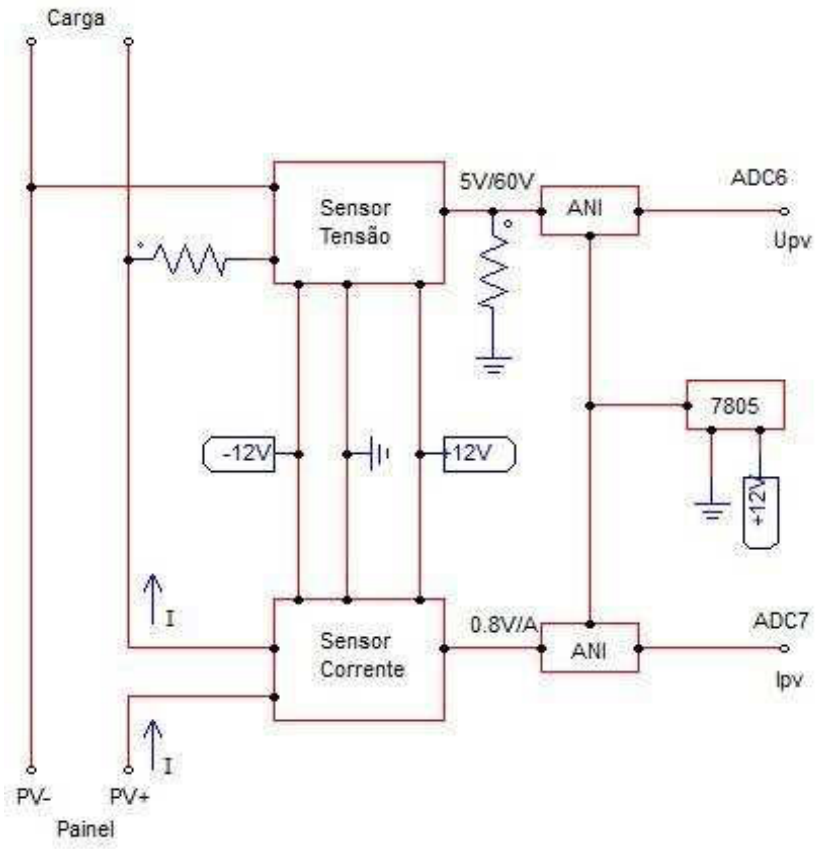
Anexos

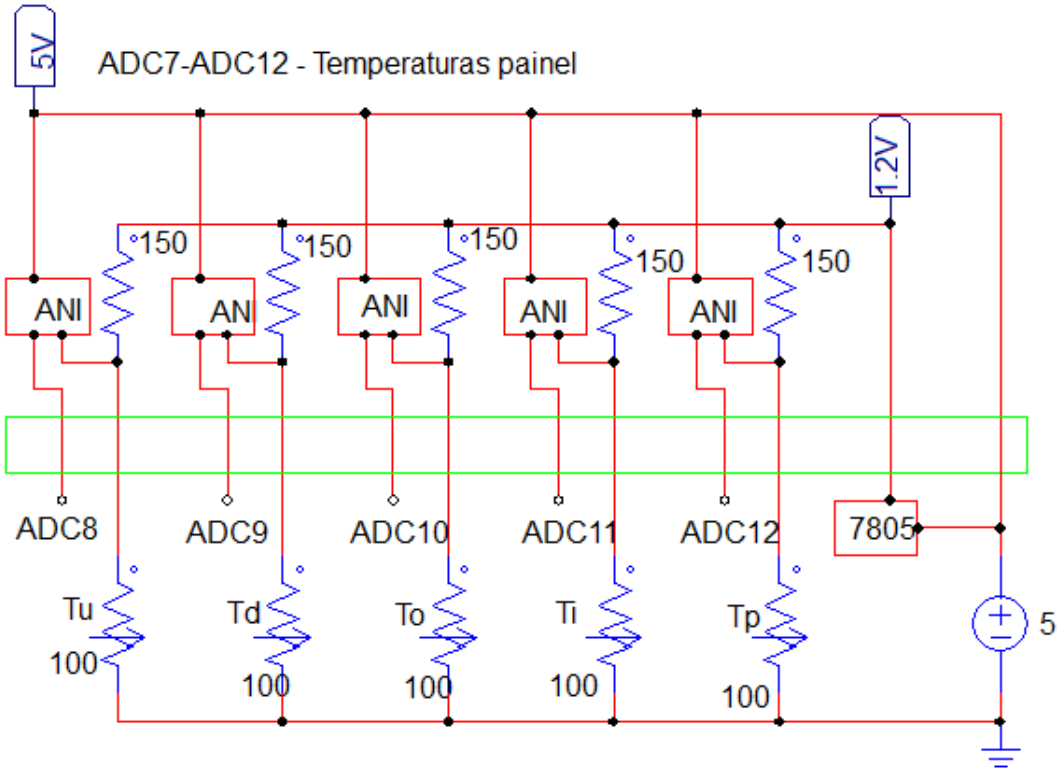
A1 – Circuitos eléctricos de condicionamento de sinal

IA - Amplificador Instrumentação
ANI - Amplificador não inversor
F2V - Conversor frequência tensão

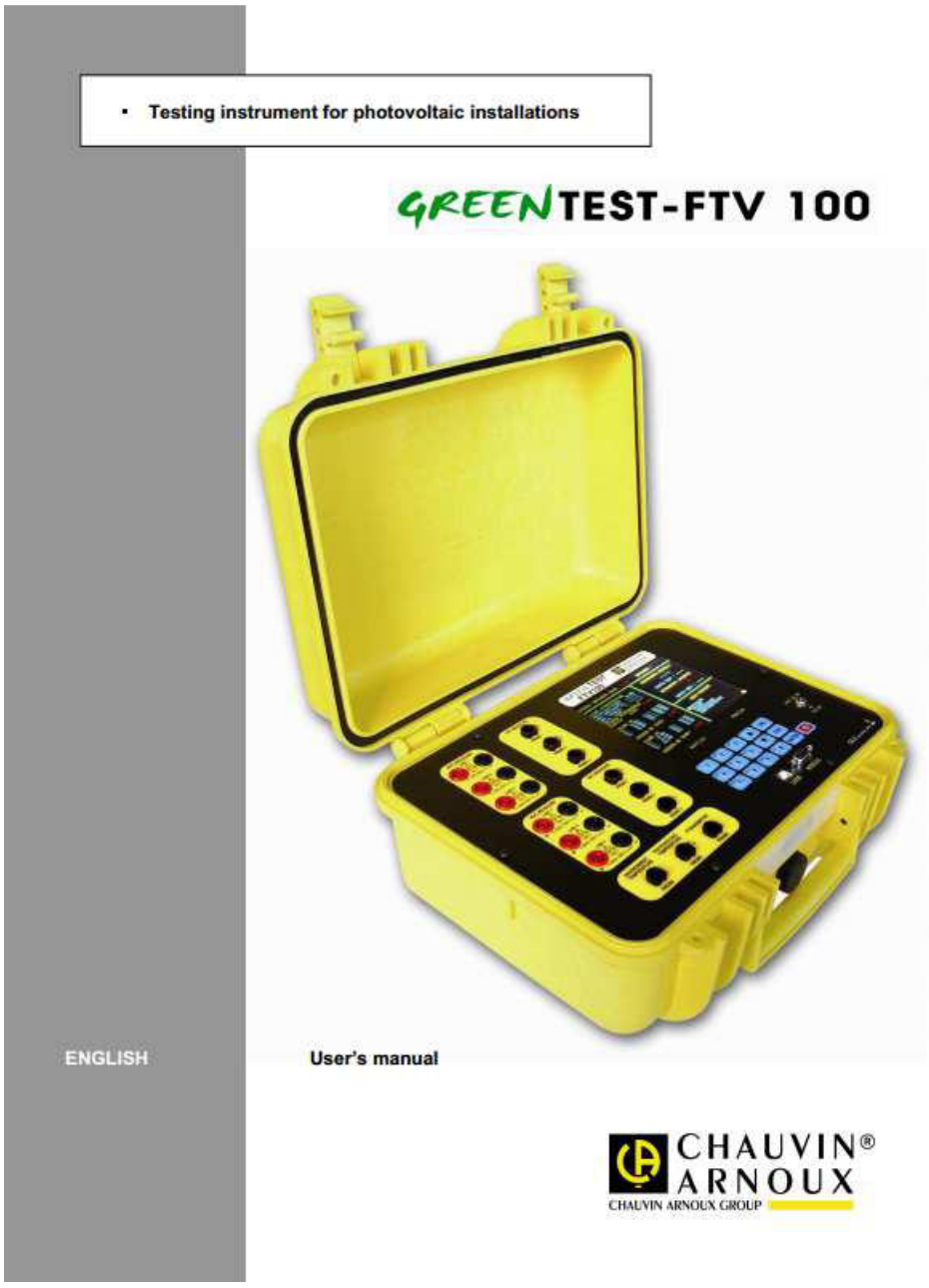
ADC1 (flow) - 4.8/138Hz(1.2l/min-0.02l/s)
ADC2 (wind speed) - 4.8/69Hz(100km/h)
ADC3 (wind direction) - 5V/360º
ADC4 (ambient temp) - 4.8/75ºC
ADC5 (radiation) - 4.8V/1250W/m2






5V
ADC7-ADC12 - Temperaturas painel
1.2V
150
150
150
150
150
ANI
ANI
ANI
ANI
ANI
ADC8
ADC9
ADC10
ADC11
ADC12
7805
Tu
Td
To
Ti
Tp
100
100
100
100
100
5

A2 – Equipamento para teste de painéis fotovoltaicos



1. input 1: ammeter clamp A_{AC}
2. input 2: ammeter clamp A_{AC}
3. input 3: ammeter clamp A_{AC}
4. input 2: ammeter clamp A_{DC}
5. input 2: ammeter clamp A_{DC}
6. input 3: ammeter clamp A_{DC}
7. input 1: voltage V_{AC}
8. input 2: voltage V_{AC}
9. input 3: voltage V_{AC}
10. input 1: voltage V_{DC}
11. input 2: voltage V_{DC}
12. input 3: voltage V_{DC}
13. ambient temperature probe input
14. pyranometer input
15. solar panel temperature probe input
16. USB serial port (to PC)
17. RS232 serial port –to remoted unit)
18. multi-function alphanumeric keypad having the following keys:
    - ▲ "up" navigation key
    - ENTER selection validation key
    - ▼ "down" navigation key
    - ⏻ On/Off button
    - ESC menu exit key
    - DEL entry deletion key
19. input of external power supply (15V DC, 2A maximum consumed)
20. PWR ON/CHARGE FULL (only when an external power supply is connected); lights during recharging of the internal batteries or when they have reached full charge.
21. BAT Ch (only when an external power supply is connected); lights during the charging of the batteries.

A3 – Fotografias dos circuitos de condicionamento com Arduino Leonardo e Arduino Yún.